PROJECT REPORT FORMAT

BYREDDY AKHILESWAR REDDY 21BCE9783

DEVALRAJU CHAITANYA 21BCE9410

MAHESH SRINIVAS UPPADA 21BCE9465

MUSUNURU SAKETH 21BCE7372

1) INTRODUCTION :

1.1) Project Overview

Crime Vision : Advanced Crime Classification

Predictive policing employs analytical methods to anticipate potential locations of future criminal activity or identify past offenders by utilizing statistical forecasts. In addressing the various aspects of a crime—such as who will commit it, who will be affected, the type of crime, location, and timing—predictive policing relies on historical data. Its primary objective in a law enforcement context is to pinpoint priority areas for resource allocation within a city. Police departments face the challenge of generating accurate crime predictions to strategically deploy patrols and resources, enhancing crime deterrence and response times.

The theoretical foundations of routine activity theory and rational choice theory contribute to understanding the conditions under which crimes occur. Routine activity theory posits that crimes happen when motivated offenders encounter suitable victims in the absence of adequate protection. Rational choice theory suggests that prospective criminals weigh the potential gains against the likelihood of being caught, making rational decisions about committing a crime. Empirical studies on near-repeat victimization reveal non-random patterns in both space and time. Traditionally, police officers analyze maps with pinned locations of reported incidents to identify and predict crime hotspots efficiently.

1.2) Purpose

Criminology literature delves into the correlation between crime and diverse factors, devising methodologies for crime prediction. A prevalent focus involves forecasting hotspots—geographical areas of varying sizes exhibiting elevated crime probabilities. Notable methods encompass Spatial and Temporal Analysis of Crime (STAC),

Thematic Mapping , and Kernel Density Estimation (KDE). STAC involves identifying the densest concentrations of points on a map, fitting them to standard deviational ellipses. Analyzing the size and alignment of these ellipses enables conclusions about the underlying nature of crime clusters.

2) Literature Survey

Paper titled "Geospatial big data handling theory and methods: A review and research challenges," Songnian Li et al. conducted a comprehensive review of geospatial theories and methods for handling large datasets. The authors highlighted the limitations of conventional data processing methods and identified areas requiring further exploration and development. Specifically, they emphasized the need for advancements in algorithms to facilitate real-time analytics and manage continuous streams of data. Additionally, the paper stressed the importance of enhancing spatial indexing techniques to cope with the unique attributes of geospatial big data.

Furthermore, the authors called for improvements in theoretical and methodological approaches for transferring big data from descriptive and parallel research and applications to those that explore casual and illustrative relationships. This underscores the importance of evolving strategies to handle geospatial big data in a manner that goes beyond conventional data processing methods.

2.1) Existing Problem

In our primary research, our focus is on identifying effective algorithms for predicting neighborhood crimes. Building on our prior work, where we utilized statistical analysis to predict crimes in New York City, our paper garnered significant attention from researchers. Motivated by this, we embarked on an exploration of efficient machine learning and deep learning approaches employed in the field of crime prediction. Employing a systematic approach, we meticulously selected papers for our review. Our research encompasses an examination of papers from diverse databases, all contributing to the exploration of innovative methods in predicting and understanding crime patterns.

2.2) Refernces

In contemporary times, the primary requirement for security revolves around automated visual surveillance. This document marks the initial stride toward automatic visual weapon identification. The paper's goal is to create a structure dedicated to the detection of weapons in visual surveillance. The suggested framework employs the K-means clustering algorithm, utilizing color-based segmentation to remove irrelevant items from an image. To pinpoint the weapon, the Speeded Up Robust Features (SURF) interest point detector is applied. The framework proves resilient against variations in scale, rotation, and occlusion. Through implementation and testing on sample weapon images, the system demonstrates efficacy under diverse image conditions.

2.3) Problem Statement Definition

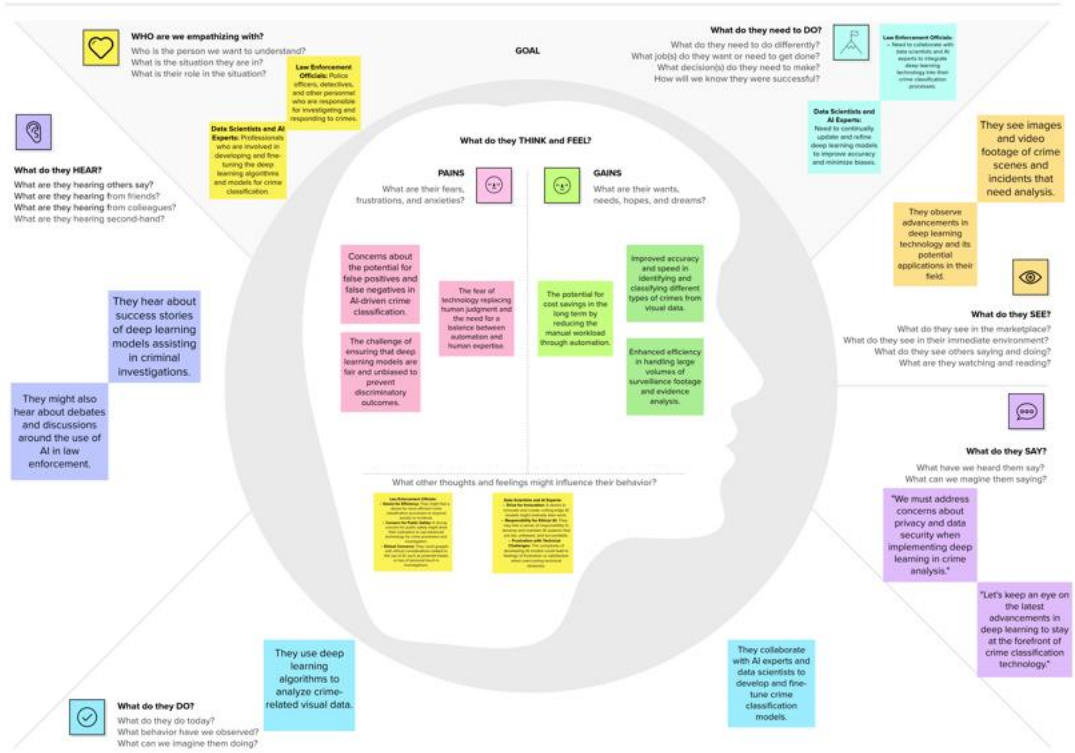
In recent times, the escalating frequency and diverse nature of crimes pose significant challenges, resulting in considerable societal losses both economically and socially. This surge in criminal activities also heightens the overall threat to the well-being of the community. Addressing this issue, the era of computing presents an opportunity not only to mitigate existing crimes but also to forecast potential criminal activities. By leveraging predictive strategies, measures can be proactively implemented to reduce the impact on property and human life. Analyzing historical data from police records, the crime rate prediction approach involves a comprehensive examination of various factors, such as the motives behind crimes, the occurrence frequency of similar incidents in specific locations, and additional relevant parameters. This multifaceted analysis contributes to the formulation of effective models for predicting crime.

3) Ideation and proposed solution

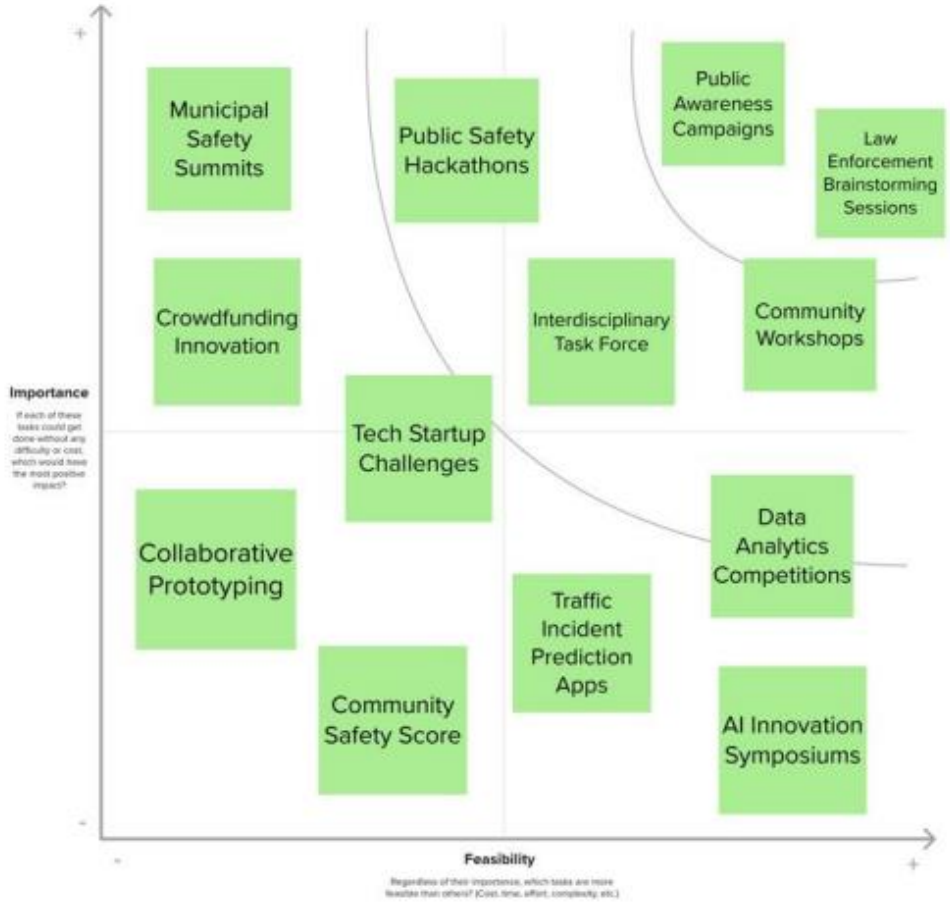
The suggested system is developed through an analysis of existing research, extensively reviewing various documents. Predominantly, crime predictions are based on the geographical location and the specific types of crimes prevalent in those regions. After a comprehensive survey of prior studies, it was observed that Linear Regression, Decision Tree, and Random Forest models consistently provide high accuracy. Consequently, these models are employed in this study

to forecast criminal activities. The dataset utilized originates from data.world.com, encompassing diverse crime types occurring in India categorized by both state and year.

3.1)Empathy map canvas



3.2)Ideation and Brain storming



4) Requirement Analysis

Requirement analysis is the process of understanding and documenting the needs and expectations of stakeholders for a particular system. For a Crime Detection and Identification System using AI/ML (Artificial Intelligence/Machine Learning) and Flask, it involves gathering and analyzing the functional and non-functional requirements to ensure the system's effectiveness.

4.1)Functional Requirements

Functional requirements specify the functions and features a system must provide. In the context of Crime Detection and Identification using AI/ML and Flask, functional requirements could include:

- Image Recognition:The system should be capable of recognizing criminal activities or individuals from images or video footage.
- Data Processing:Efficient processing of large datasets containing crime-related information.

- Crime Classification:Classification of crimes based on the detected patterns, providing insights into the nature of incidents.
- Real-time Monitoring:Continuous monitoring of live feeds for immediate crime detection.
- Alert Generation:Automated alerts or notifications to law enforcement agencies upon the identification of suspicious activities.

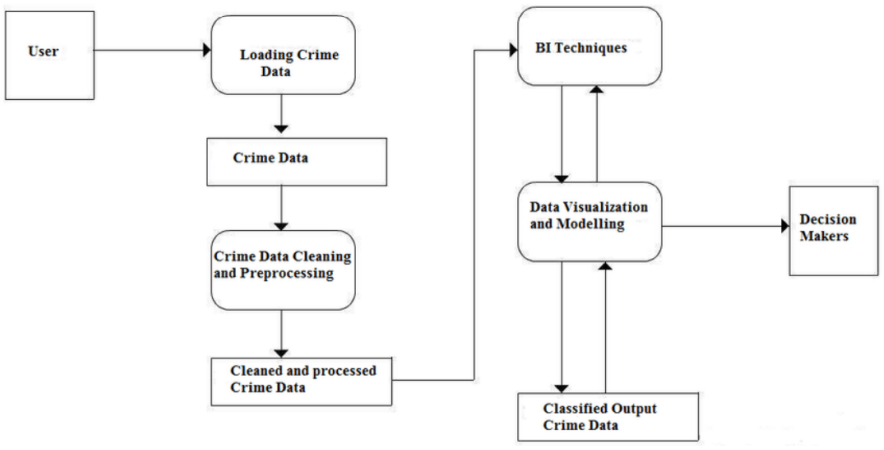
4.2)Non Functional Requirements

Non-functional requirements define system attributes such as performance, usability, reliability, and security. For a Crime Detection System, non-functional requirements could encompass:

- Accuracy:The system should achieve a high level of accuracy in crime detection to minimize false positives and negatives.
- Scalability:The ability to handle an increasing volume of data and user interactions without compromising performance.
- Response Time:Real-time responsiveness for timely intervention in criminal activities.
- Security:Ensuring the confidentiality and integrity of sensitive crime data.
- Usability:A user-friendly interface for law enforcement personnel to interact with the system efficiently.
- Reliability:Dependability and consistency in crime detection and identification.

5) Project Design

5.1)Data flow diagram and user stories

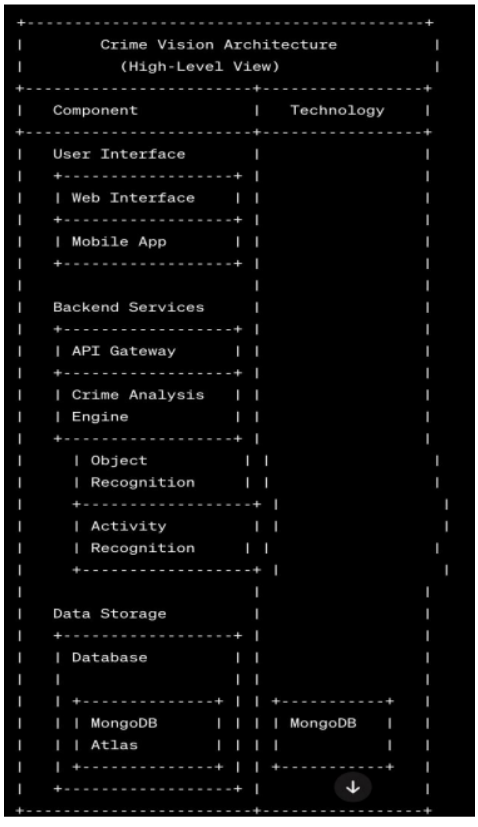


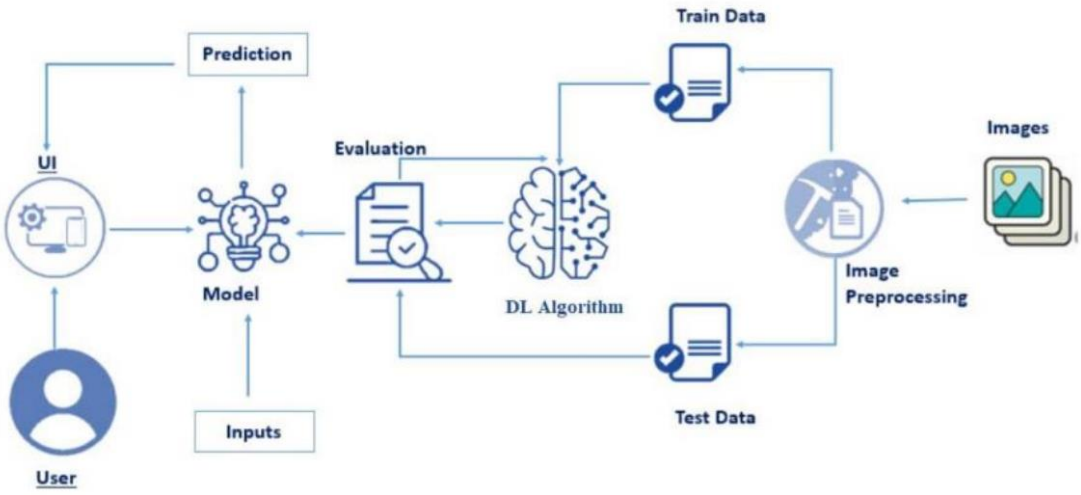
User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user) Customer	Registration & Access	USN-1	As a law enforcement officer, I can register for the Crime Classification System by providing necessary credentials.	I can log into the Crime Classification System after registration.	High	Sprint-1
(Web user)						
	Real-time Crime Analysis	USN-2	As a law enforcement officer, I can submit real-time crime data for analysis using the deep learning model.	I receive accurate crime classification results promptly.	High	Sprint-1
	Historical Crime Analysis	USN-3	As a law enforcement officer, I can access historical crime data for analysis and investigation.	The system displays relevant historical crime information based on specified criteria.	Medium	Sprint-2
	Notification Alerts	USN-4	As a law enforcement officer, I can receive real-time notification alerts for emerging criminal patterns.	I receive timely alerts for potential criminal activities.	High	Sprint-1
Administrator	User Management	USN-5	As a system administrator, I can manage user accounts, including creating, modifying, and deactivating accounts.	User accounts are created, modified, or deactivated as per administrator actions.	High	Sprint-1
	Model Training & Update	USN-6	As a system administrator, I can initiate the training of the crime classification model and update it with new data.	The model is successfully trained and updated without errors.	High	Sprint-2
	System Monitoring	USN-7	As a system administrator, I can monitor the system's performance, including resource usage and response times.	The administrator receives real-time information on system performance.	Medium	Sprint-2

5.2)Solution Architecture





6) Project Planning and Scheduling

6.1) Technical Architecture

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	UI (User Interface)	The system incorporates visual and interactive components that enable users to engage with and manipulate its features.	HTML, CSS, JavaScript, UI frameworks (e.g., React, Angular)
2.	Model	The computational framework or model that acquires knowledge from data and generates forecasts.	Deep learning frameworks (e.g., TensorFlow, PyTorch), programming languages (e.g., Python), neural network architectures (e.g., CNN, RNN)
3.	Deep Learning Algorithm	The particular algorithm or method employed in the deep learning model to discern patterns and make forecasts.	Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), YOLO (You Only Look Once), etc.
4.	Evaluation	Evaluating the model's performance and its effectiveness in action.	Metrics (e.g., accuracy, precision, recall, F1score), programming languages (e.g., Python), data analysis libraries (e.g., NumPy, pandas)
5.	Image Pre-processing	The input images undergo various operations to improve their quality and make them more suitable for analysis..	Image processing libraries (e.g., OpenCV), programming languages (e.g., Python), image manipulation techniques (e.g., resizing, normalization)
6.	Train Data	The data with assigned labels utilized for training the deep learning model.	Labelled image datasets, data collection and labelling tools
7.	Test Data	The labeled dataset employed to assess the effectiveness of the trained model..	Labelled image datasets, data collection and labelling tools

Table-2: Application Characteristics:

S.no	Characteristic	Description	Technology
1.	Open-Source Frameworks	Utilizing open-source frameworks with publicly accessible source code that enables customization and fosters collaboration within the community.	TensorFlow, PyTorch, Keras, scikitlearn, OpenCV, Django, Flask, Node.js
2.	Security Implementations	Incorporating safeguards to secure data and guarantee the system's data confidentiality, integrity, and availability.	Encryption algorithms (e.g., AES, RSA), secure communication protocols (e.g., SSL/TLS), authentication mechanisms, access control systems

6.2) Sprint Planning and Estimation

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the Crime Vision application by providing necessary details	2	High	Akhileswar, Chaitanya, Mahesh, Musunuru
Sprint-1		USN-2	As a user, I will receive a confirmation email once I have registered for the Crime Vision application	1	High	Akhileswar, Chaitanya, Mahesh, Musunuru
Sprint-2		USN-3	As a user, I can register for the Crime Vision application through Facebook.	2	Low	Chaitanya, Akhileswar
Sprint-1		USN-4	As a user, I can register for the Crime Vision application through Gmail.	2	Medium	Akhileswar, Chaitanya, Mahesh, Musunuru
Sprint-1	Login	USN-5	As a user, I can log into the Crime Vision application by entering my credentials.	1	High	Akhileswar, Chaitanya, Mahesh, Musunuru
	Dashboard					

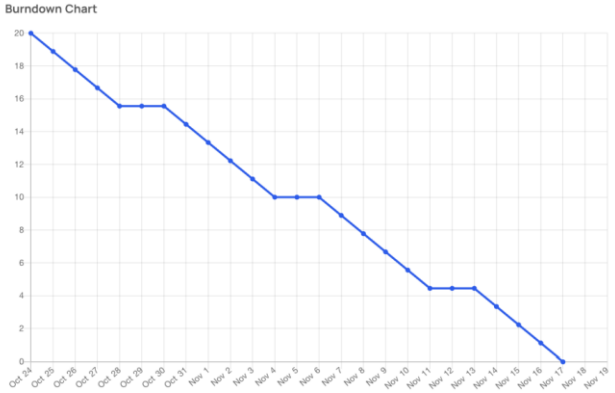
6.3) Sprint delivery schedule

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 2023	29 Oct 2023	20	29 Oct 2023
Sprint-2	20	6 Days	31 Oct 2023	05 Nov 2023	20	05 Nov 2023
Sprint-3	20	6 Days	07 Nov 2023	12 Nov 2023	20	12 Nov 2023
Sprint-4	20	6 Days	14 Nov 2023	19 Nov 2023	20	19 Nov 2023

Velocity: Imagine we have a 6-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)
Velocity=20/6

Burndown Chart:

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.



7) Coding and Solutioning

Project Flow:

- The user interacts with the UI to choose an image.
- The chosen image is processed by a transfer learning deep learning model.
- The transfer learning model is integrated with a Flask application.
- The transfer learning model analyzes the image and generates predictions.
- The predictions are displayed on the Flask UI for the user to see.
- This process enables users to input an image and receive accurate predictions quickly.

To accomplish this, we have to complete all the activities and tasks listed below ▪ Data Collection.

o Create a Train and Test path.

- Image Pre-processing. o Import the required library o Configuration of Images and preprocessing o

Apply Image_Dataset_from_directory functionality to Train set and Test set

- Model Building o Create Transfer Learning Function o Adding Dense Layer o

Configure the Learning Process o Train the model o Save the Model o Test the model

- Application Building o Create an HTML file o Build Python Flask Code

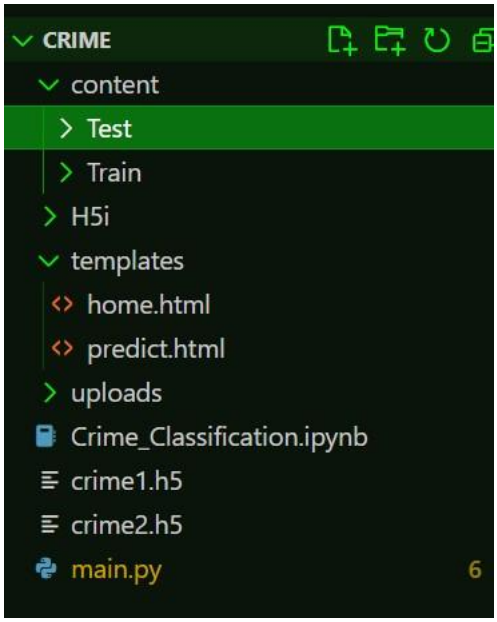
Prior Knowledge:

You must have prior knowledge of following topics to complete this project. **Deep Learning Concepts**

- **CNN:** https://towardsdatascience.com/basics-of-the-classic-cnna3dce1225add
- **TransferLearning:**https://www.analyticsvidhya.com/blog/2021/10/under standingtransfer-learning-for-deep-learning/
- **Flask:** Flask is a popular Python web framework, meaning it is a third-party Python library used for developing web applications.
  Link: https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Structure:

Create a Project folder which contains files as shown below



- The Dataset folder contains the training and testing images for training our model.
- For building a Flask Application we needs HTML pages stored in the **templates** folder and a python script **main.py** for server side scripting.
- We also have Crime_Classification.ipynb file and crime1.h5 in which the model is saved.

Milestone 1: Data Collection

There are many popular open sources for collecting the data.
Example : kaggle.com, UCI repository, etc.

Activity 1: Download the dataset

The dataset contains images extracted from every video from the UCF Crime Dataset. Every 10th frame is extracted from each full-length video and combined for every video in that class. All the images are of size 64*64 and in .png format

The dataset has a total of 14 Classes :

You can download the dataset used in this project

using the below link Dataset:

https://www.kaggle.com/datasets/odins0n/ucf-crimedataset

Note: For better accuracy train on more images

We are going to build our training model on Google colab so we have to upload a dataset zip file on Google colab. To upload a dataset zip file to Google Colab and then unzip it, you

can follow these steps:

- Open Google Colab and create a new notebook.
- Click on the "Files" icon on the left-hand side of the screen.
- Click on the "Upload" button and select the zip file you want to upload.
- Wait for the upload to complete. You should see the file appear in the "Files" section.
- To unzip the file, you can use the following command:

```
!unzip /content/ucf-crime-dataset.zip

Streaming output truncated to the last 5000 lines.
  inflating: Train/Vandalism/Vandalism035_x264_230.png
  inflating: Train/Vandalism/Vandalism035_x264_240.png
  inflating: Train/Vandalism/Vandalism035_x264_250.png
  inflating: Train/Vandalism/Vandalism035_x264_260.png
  inflating: Train/Vandalism/Vandalism035_x264_270.png
  inflating: Train/Vandalism/Vandalism035_x264_280.png
  inflating: Train/Vandalism/Vandalism035_x264_290.png
  inflating: Train/Vandalism/Vandalism035_x264_30.png
  inflating: Train/Vandalism/Vandalism035_x264_300.png
  inflating: Train/Vandalism/Vandalism035_x264_310.png
  inflating: Train/Vandalism/Vandalism035_x264_320.png
  inflating: Train/Vandalism/Vandalism035_x264_330.png
  inflating: Train/Vandalism/Vandalism035_x264_340.png
  inflating: Train/Vandalism/Vandalism035_x264_350.png
  inflating: Train/Vandalism/Vandalism035_x264_360.png
  inflating: Train/Vandalism/Vandalism035_x264_370.png
  inflating: Train/Vandalism/Vandalism035_x264_380.png
  inflating: Train/Vandalism/Vandalism035_x264_390.png
  inflating: Train/Vandalism/Vandalism035_x264_40.png
  inflating: Train/Vandalism/Vandalism035_x264_400.png
  inflating: Train/Vandalism/Vandalism035_x264_410.png
  inflating: Train/Vandalism/Vandalism035_x264_420.png
  inflating: Train/Vandalism/Vandalism035_x264_430.png
```

Activity 2: Create training and testing dataset

To build a DL model we have to split training and testing data into two separate folders. But In the project dataset folder training and testing folders are presented. So, in this case we just have to assign a variable and pass the folder path to it.

```
train_dir="/content/Train"
test_dir="/content/Test"
```

Milestone 2: Image Preprocessing

In this milestone we will be improving the image data that suppresses unwilling distortions or enhances some image features important for further processing, although performing some geometric transformations of images like rotation, scaling, translation, etc.

Activity 1: Importing the libraries

Import the necessary libraries as shown in the image

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import os

import tensorflow as tf
from tensorflow.keras.preprocessing import image_dataset_from_directory

from tensorflow.keras.applications import DenseNet121
from sklearn.preprocessing import LabelBinarizer
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout,MaxPooling2D , Conv2D,Fl
from tensorflow.keras.models import Sequential

from sklearn.metrics import roc_curve, auc, roc_auc_score
from sklearn.metrics import classification_report


from IPython.display import clear_output
import warnings
warnings.filterwarnings('ignore')
```

To understand the above imported libraries:-

Image_dataset_from_directory : is a function in the tensorflow.keras.preprocessing module of TensorFlow, which allows you to create a TensorFlow Dataset from a directory containing image files. This function can be useful for training deep learning models on large image datasets.
Keras: Keras is a high-level neural network API written in Python that allows for fast experimentation and prototyping of deep learning models.
DenseNet121:It has been trained on large-scale image classification datasets such as ImageNet and has achieved state-of-the-art performance on a range of benchmark tasks. It has also been used as a pre-trained model for transfer learning in various computer vision applications.

Global average pooling 2D (GAP 2D): It is a type of pooling operation commonly used in convolutional neural networks (CNNs) for image classification tasks.

Dense Layer: A dense layer in neural networks is a fully connected layer where each neuron in the layer is connected to every neuron in the previous layer, and each connection has a weight associated with it.

Flatten Layer: A flatten layer in neural networks is a layer that reshapes the input tensor into a one-dimensional array, which can then be passed to a fully connected layer.

Input Layer: The input layer in neural networks is the first layer of the network that receives the input data and passes it on to the next layer for further processing. **Maxpooling 2D :** It is a Downsampling operation that reduces the spatial dimensions (height and width) of an input tensor while preserving the number of channels. The operation takes a window of a fixed size, typically 2x2, and outputs the maximum value within that window. Max Pooling helps reduce the computational cost of the network while also increasing its robustness to small translations of the input.

Convolution 2D: It is a linear operation that applies a set of learnable filters (also called kernels or weights) to an input tensor to extract features. The filters slide over the input tensor, computing a dot product between their values and the values of the input tensor at each position. The output of a convolutional layer is a set of feature maps, each corresponding to a specific filter. Convolution 2D is the main building block of CNNs and is used to learn representations of the input data.

image: from tensorflow.keras.preprocessing import image imports the image module from Keras' tensorflow.keras.preprocessing package. This module provides a number of image preprocessing utilities, such as loading images, converting images to arrays, and applying various image transformations.
load_img: load_img is a function provided by the

tensorflow.keras.preprocessing.image module that is used to load an image file from the local file system. It takes the file path as input and returns a PIL (Python Imaging Library) image object.

Dropout : is a regularization technique that randomly drops out a fraction of the neurons in a neural network during training. This helps to prevent overfitting, which is when a model performs well on the training data but poorly on new data. Dropout forces the network to learn more robust

features by preventing any one neuron from becoming too important in the network's predictions.

Numpy: It is for performing mathematical functions

Matplotlib: Matplotlib is a data visualization library in Python that is widely used for creating high-quality, publication-ready plots and charts.

clear_output: command is used to clear the output of a Jupyter notebook cell. This can be useful when you want to update the output of a cell with new information, or when you want to remove previous output that is no longer relevant.

Activity 2: Configuration of Images and preprocessing

```
SEED = 12
IMG_HEIGHT = 64
IMG_WIDTH = 64
BATCH_SIZE = 128
EPOCHS = 1
LR =  0.00003
```

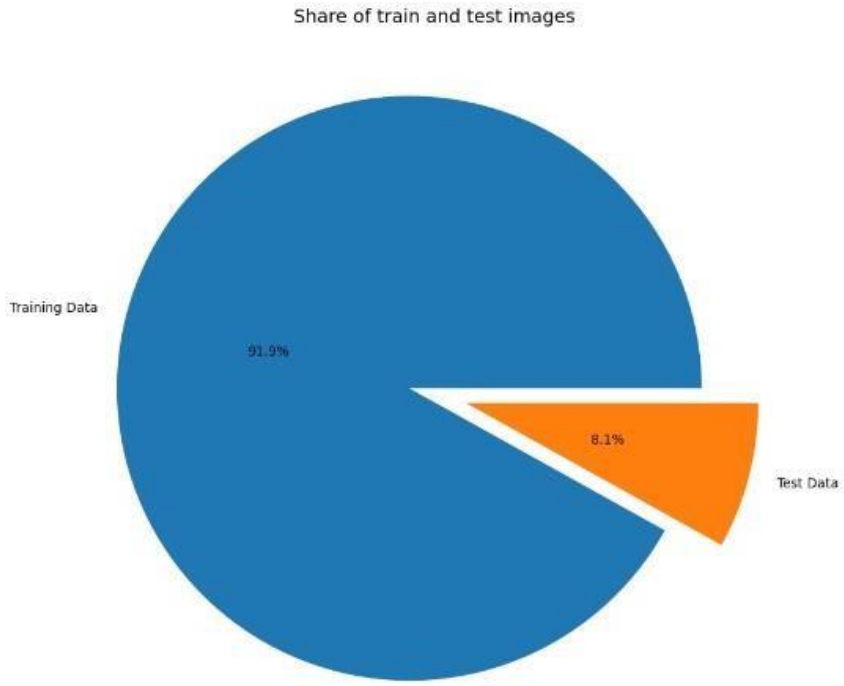
directory: Directory where the data is located. If labels are "inferred", it should contain subdirectories, each containing images for a class. Otherwise, the directory structure is ignored. **batch size**: Size of the batches of data which is 64. **target size**: Size to resize images after they are read from disk.

Learning Rate (LR): The learning rate is a hyperparameter that determines the step size at each iteration during gradient descent optimization. Gradient

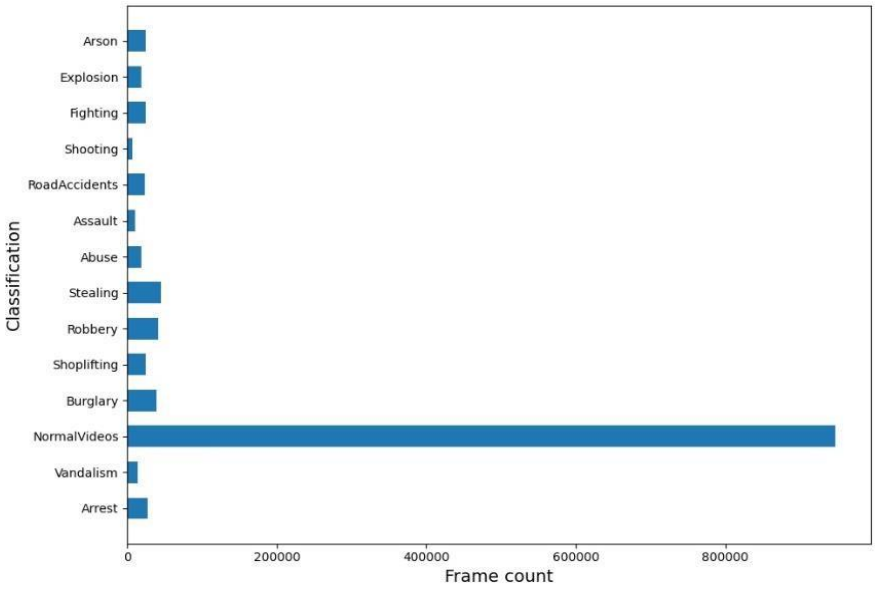
descent is the most common optimization algorithm used in machine learning o update the weights of a neural network during training. The learning rate controls how quickly or slowly the weights are updated in response to the error gradient. A high learning rate can cause the algorithm to converge too quickly, whereas a low learning rate can cause slow convergence or even prevent the model from converging. **Seeds**: Seeds are used in machine learning to ensure that results are reproducible. A seed is a random number that is used to initialize the random number generator before training the model. By setting the seed value, we can ensure that the same sequence of random numbers is generated every time the code is run. This is important when developing models as it allows us to compare results between different runs and ensure that any changes to the model or hyperparameters are actually improving performance. Now it is time to Build input and output layers for Transfer Learning model Hidden layers freeze because they have trained sequence, so changing the input and output layers.

```
crimes={}
train=test=0
for clss in crime_types:
    num=len(os.listdir(os.path.join(train_dir,clss)))
    train+=num
    test+=len(os.listdir(os.path.join(test_dir,clss)))

    crimes[clss]=num
```

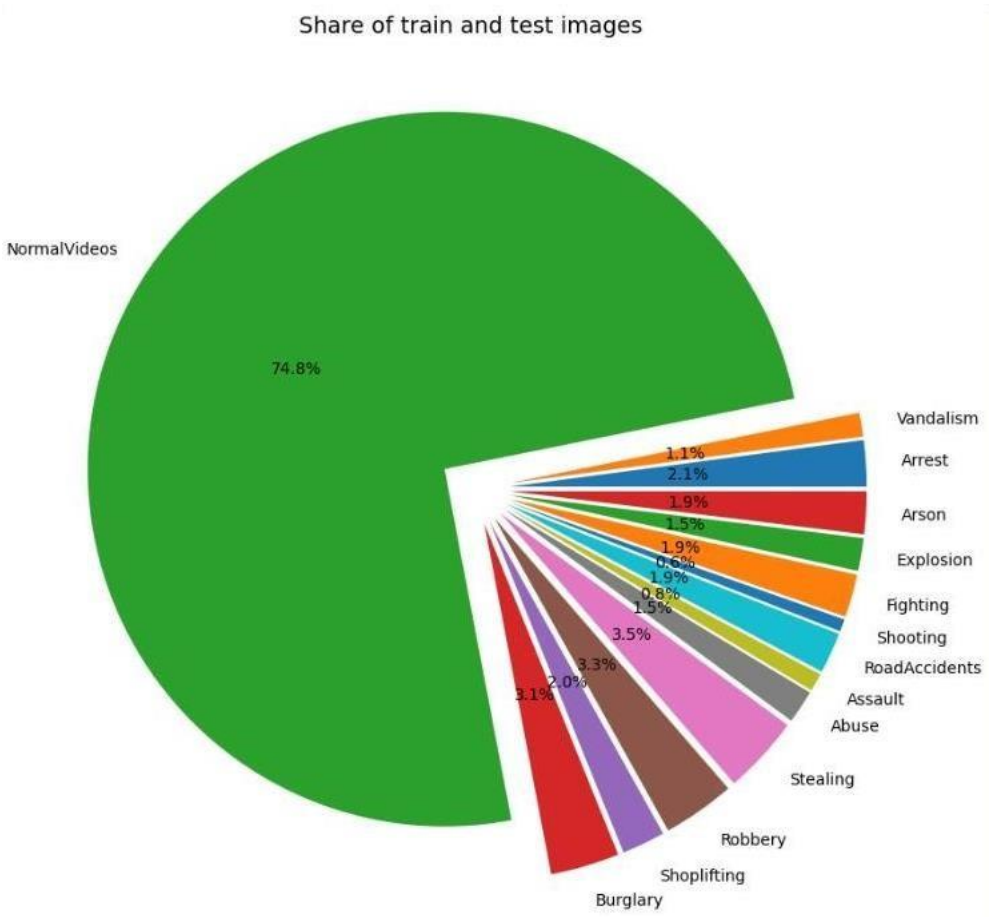
```
plt.figure(figsize=(15, 10))
plt.pie(x=np.array([train,test]), autopct="%.1f%%", explode=[0.1, 0.1],
        labels=["Training Data", "Test Data"], pctdistance=0.5)
plt.title("Share of train and test images ", fontsize=14);
```




```
plt.figure(figsize=(10, 7))
plt.barh(list(crimes.keys()), list(crimes.values()), height=0.6, align="center")
plt.yticks(rotation=0)

plt.xlabel("Frame count", fontsize=14)
plt.ylabel("Classification", fontsize=14)
plt.tight_layout()
plt.show()
```



```
plt.figure(figsize=(15,10))
plt.pie(x=np.array(list(crimes.values())), autopct="%.1f%%",
        explode=[0.1]*n,labels=list(crimes.keys()), pctdistance=0.5)
plt.title("Share of train and test images ", fontsize=14);
```



Activity 3: Apply Image_Dataset_from_directory functionality to Train set and Test set

ImageDataset.from_directory() is a function from the TensorFlow library used to load images from a directory and create a dataset object that can be used for machine learning tasks, such as image classification or object detection.

The function takes the following arguments:

- **directory**: A string representing the directory where the images are located.
- **labels**: A list of strings representing the class labels for the images.
- **batch_size**: An integer representing the number of images to load in each batch.
- **image_size**: A tuple representing the size to which the images should be resized.

```
train_set=image_dataset_from_directory(
    train_dir,
    label_mode="categorical",
    batch_size=BATCH_SIZE,
    image_size=IMG_SHAPE,
    shuffle=True,
    seed=seed,
    validation_split=0.2,
    subset="training",
)

val_set=image_dataset_from_directory(
    train_dir,
    label_mode="categorical",
    batch_size=BATCH_SIZE,
    image_size=IMG_SHAPE,
    shuffle=True,
    seed=seed,
    validation_split=0.2,
    subset="validation",
)
```

```
test_set=image_dataset_from_directory(
    test_dir,
    label_mode="categorical",
    class_names=None,
    batch_size=BATCH_SIZE,
    image_size=IMG_SHAPE,
    shuffle=False,
    seed=seed,
)
```

```
Found 1266345 files belonging to 14 classes.
Using 1013076 files for training.
Found 1266345 files belonging to 14 classes.
Using 253269 files for validation.
Found 111308 files belonging to 14 classes.
```

Milestone 3: Model Building

Activity 1: Create Transfer Learning Function

Now, let us create transfer learning function with DenseNet121 with parameters include_top, and weights imagenet with mentioned input shape. Also, we are setting threshold at 149.

DenseNet is a convolutional neural network where each layer is connected to all other layers
that are deeper in the network, that is, the first layer is connected
to the 2nd, 3rd, 4th and so on, the second layer is connected to
the 3rd, 4th, 5th and so on

```
def transfer_learning():
    base_model=DenseNet121(include_top=False,input_shape=INPUT_SHAPE,weights="imagenet")

    thr=149
    for layers in base_model.layers[:thr]:
        layers.trainable=False

    for layers in base_model.layers[thr:]:
        layers.trainable=True

    return base_model
```

include_top: whether to include the fully-connected layer at the top of the network. **weights**: one of None (random initialization), 'imagenet' (pre-training on ImageNet), or the path
to the weights file to be loaded. **input_shape**: optional shape tuple, only to be specified if include_top is False (otherwise the
input shape has to be (224, 224, 3).

Activity 2:

Adding Dense Layers

A **dense** layer is a deeply connected neural network layer. It is the most common and frequently used layer.The number of neurons in the Dense layer is the same as the number of classes in the training set. The neurons in the last Dense layer, use softmax activation to convert their outputs into respective

probabilities. Understanding the model is a very important phase to properly use it for training and prediction purposes.

```
def create_model():
    model=Sequential()

    base_model=transfer_learning()
    model.add(base_model)

    model.add(GlobalAveragePooling2D())

    model.add(Dense(128, activation="relu"))
    model.add(Dropout(0.2))

    model.add(Dense(n,activation="softmax",name="classification"))

    model.summary()

    return model
```

Keras provides a simple method, summary to get the full information about the model and its layers.

```
model=create_model()

model.compile(optimizer="adam",
              loss='categorical_crossentropy',
              metrics = [tf.keras.metrics.AUC()])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applicat
29084464/29084464 [==============================] - 0s 0us/step
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 densenet121 (Functional)    (None, 2, 2, 1024)        7037504

 global_average_pooling2d (  (None, 1024)              0
 GlobalAveragePooling2D)

 dense (Dense)               (None, 128)               131200

 dropout (Dropout)           (None, 128)               0

 classification (Dense)      (None, 14)                1806

=================================================================
Total params: 7170510 (27.35 MB)
Trainable params: 5586254 (21.31 MB)
Non-trainable params: 1584256 (6.04 MB)
_________________________________________________________________
```

Activity 3: Configure the Learning Process

The compilation is the final step in creating a model. Once the compilation is done, we can move on to the training phase. The loss function is used to find errors or deviations in the learning process. Keras requires a loss function during the model compilation process.

Optimization is an important process that optimizes the input weights by comparing the prediction and the loss function. Here we are using adam optimizer

```
model=create_model()

model.compile(optimizer="adam",
              loss='categorical_crossentropy',
              metrics = [tf.keras.metrics.AUC()])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applic
29084464/29084464 [==============================] - 0s 0us/step
Model: "sequential"
```

Metrics are used to evaluate the performance of your model. It is similar to the loss function, but not used in the training process.

Activity 4: Train the model

Now, let us train our model with our image dataset. The model is trained for 5 epochs and after every epoch, the current model state is saved if the model has the least loss encountered till that time. We can see that the training loss decreases in almost every epoch.

fit_generator functions used to train a deep learning neural network

Arguments:

- **steps_per_epoch:** it specifies the total number of steps taken from the generator as soon as one epoch is finished and the next epoch has started. We can calculate the value of steps_per_epoch as the total number of samples in your dataset divided by the batch size.
- **Epochs:** an integer and number of epochs we want to train our model for.
- **validation_data can be either:**
- an inputs and targets list
- a generator
- an inputs, targets, and sample_weights list which can be used to evaluate the loss and metrics for any model after any epoch has ended.

- **validation_steps:** only if the validation_data is a generator then only this argument can be used. It specifies the total number of steps taken from the generator before it is stopped at every epoch and its value is calculated as the total number of validation data points in your dataset divided by the validation batch size.

```
history = model.fit(x = train_set,validation_data=val_set,epochs = 5)


Epoch 1/5
7915/7915 [==============================] - 1698s 205ms/step - loss: 0.0610 - auc: 0.9991 - val_loss: 0.0180 - val_auc: 0.999
Epoch 2/5
7915/7915 [==============================] - 1535s 194ms/step - loss: 0.0179 - auc: 0.9997 - val_loss: 0.0114 - val_auc: 0.999
Epoch 3/5
7915/7915 [==============================] - 1547s 195ms/step - loss: 0.0121 - auc: 0.9998 - val_loss: 0.0079 - val_auc: 0.999
Epoch 4/5
7915/7915 [==============================] - 1497s 189ms/step - loss: 0.0097 - auc: 0.9999 - val_loss: 0.0071 - val_auc: 0.999
Epoch 5/5
7915/7915 [==============================] - 1457s 184ms/step - loss: 0.0084 - auc: 0.9999 - val_loss: 0.0077 - val_auc: 0.999
```

Accuracy of the model after 5 epochs.

Milestone 4: Save the Model

The model is saved with .h5 extension as follows An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.

```
model.save("crime1.h5")
```

Testing the model:

Evaluation is a process during the development of the model to check whether the model is the best fit for the given problem and corresponding data. Load the saved model using load_model

```
from tensorflow.keras.models import load_model
model.load_weights('/content/crime1.h5')



y_true= np.array([])
for x,y in test_set:
  y_true=np.concatenate([y_true,np.argmax(y.numpy(),axis=-1)])
```

```
  y_true


array([ 0.,  0.,  0., ..., 13., 13., 13.])



  import keras.utils as image
  from tensorflow.keras.preprocessing.image import load_img
```

Taking an image as input and checking the results

```
  from tensorflow.keras.preprocessing import image
  img= image.load_img("/content/Test/Abuse/Abuse028_x264_1070.png",target_size=(64,64))
  x=image.img_to_array(img)
  x=np.expand_dims(x,axis=0)
  pred=np.argmax(model.predict(x))
  op=['Fighting','Burglary','Vandalism','Assault','Stealing','RoadAccidents','NormalVideos'
     ,'Explosion','Abuse','Robbery','Arrest','Shooting','Shoplifting','Arson']
  print(pred)
  op[pred]


1/1 [==============================] - 7s 7s/step
8

'Abuse'
```

```
from tensorflow.keras.preprocessing import image
img= image.load_img("/content/Test/Arson/Arson007_x264_1110.png",target_size=(64,64))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
pred=np.argmax(model.predict(x))
op=['Fighting','Burglary','Vandalism','Assault','Stealing','RoadAccidents','NormalVideos',
    'Explosion','Abuse','Robbery','Arrest','Shooting','Shoplifting','Arson']
print(pred)
op[pred]
```

```
1/1 [==============================] - 0s 284ms/step
13

'Arson'
```

```
img= image.load_img("/content/Test/Arson/Arson007_x264_1140.png",target_size=(64,64))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
pred=np.argmax(model.predict(x))
op=['Fighting','Burglary','Vandalism','Assault','Stealing','RoadAccidents','NormalVideos'
    ,'Explosion','Abuse','Robbery','Arrest','Shooting','Shoplifting','Arson']
print(pred)
op[pred]
```

```
1/1 [==============================] - 0s 149ms/step
13

'Arson'
```

Milestone 5: Application Building

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI. This section has the following tasks

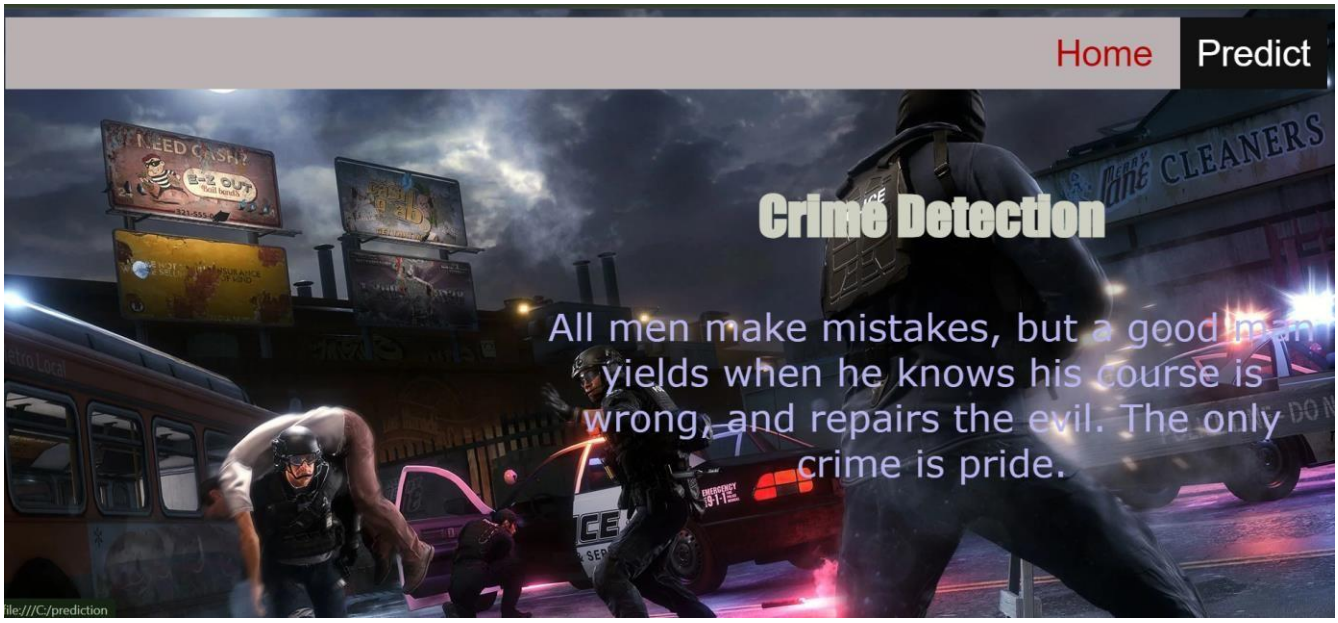
- Building HTML Pages
- Building python code

Activity1: Building Html Pages:

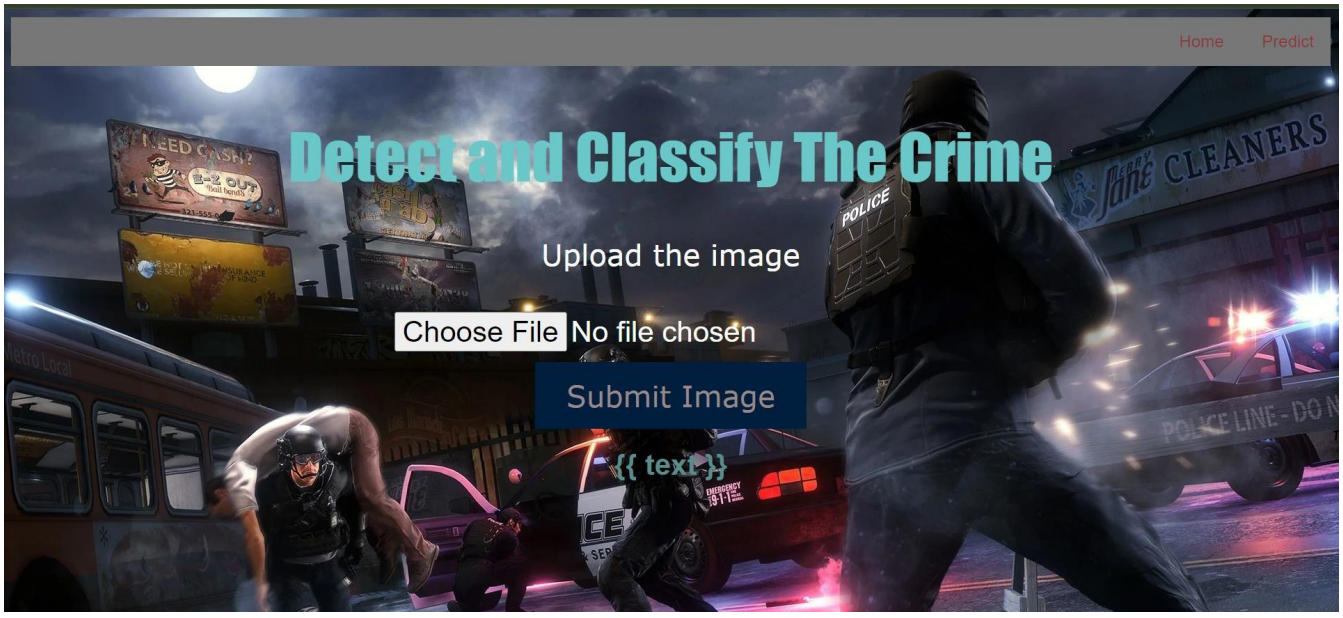
For this project create one HTML file namely

- home.html
- predict.html

Let's see how our home.html page looks like:



When you click on the predict button, you will be redirecting to the following page



When you click on the Choose button, it will redirect you to the below page.

From here you can choose different images for getting prediction

You will get the prediction in the same prediction page.

Activity 2: Build Python code:

Import the libraries

```
import os
import numpy as np
from flask import Flask, request, render_template, redirect
import tensorflow as tf
from keras.models import load_model
from keras.utils import load_img, img_to_array
from werkzeug.utils import secure_filename
```

Loading the saved model and initializing the flask app

```
app = Flask(__name__)

model = load_model("crime1.h5", compile=False)
```

Render HTML pages:

```
# home page
@app.route('/')
def home():
    return render_template('home.html')


# prediction page
@app.route('/prediction')
def prediction():
    return render_template('predict.html')
```

Once we uploaded the file into the app, then verifying the file uploaded properly or not. Here we will be using declared constructor to route to the HTML page which we have created earlier.

In the above example, '/' URL is bound with prediction.html function. Hence, when the home page of the web server is opened in browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:

```
@app.route('/predict', methods=['POST'])
def predict():
    if request.method == 'POST':
        f = request.files['image']
        base_path = os.path.dirname(__file__)
        file_path = os.path.join(base_path, 'uploads', secure_filename(f.filename))
        f.save(file_path)
        img = load_img(file_path, target_size=(64, 64))
        x = img_to_array(img)
        x = np.expand_dims(x, axis=0)
        pred = np.argmax(model.predict(x), axis=1)
        op = ['Abuse', 'Arrest', 'Arson', 'Assault', 'Burglary', 'Explosion', 'Fighting', 'Stealing',
              'Vandalism', 'Robbery', 'Shooting', 'Nrml', 'Shoplifting', 'RoadAccidents']
        result = 'The predicted output is '+(str(op[pred[0]]))
    return render_template('predict.html', text=result)

if __name__ == "__main__":
    app.run(debug=True)
```

Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model. Predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier

Main Function:

```
if __name__ == "__main__":
    app.run(debug=True)
```

Activity 3: Run the application

Open anaconda prompt from the start menu

Navigate to the folder where your python script is.

Now type "python app.py" command

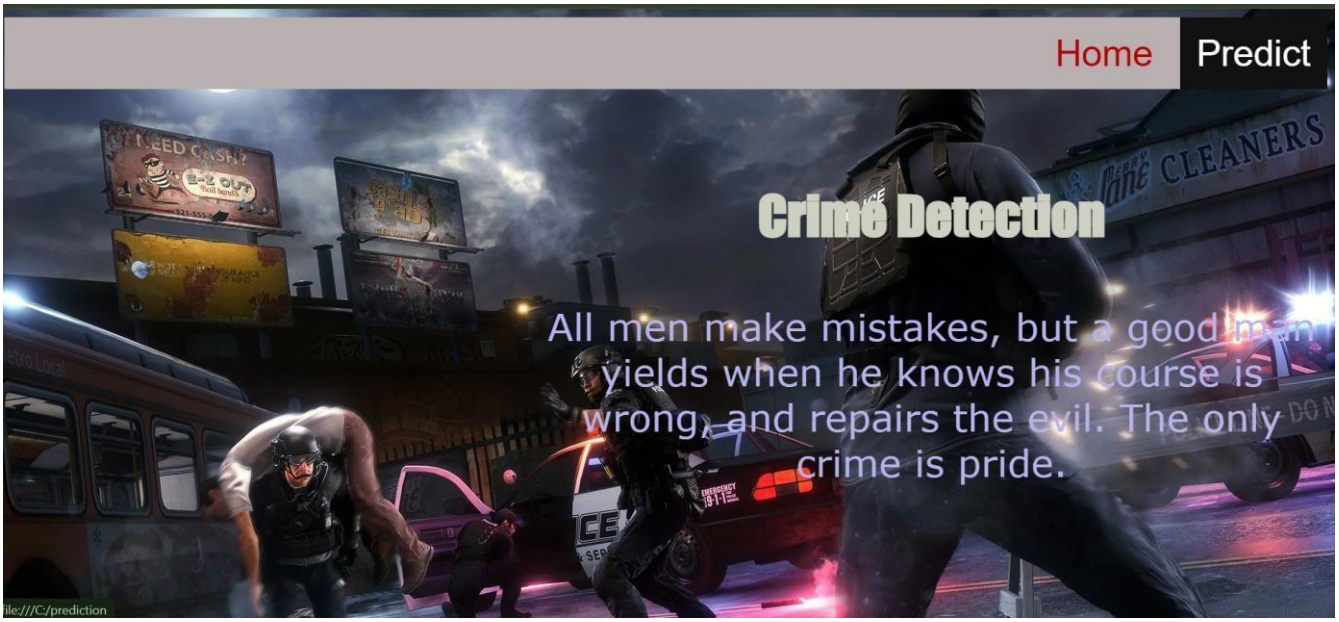
Navigate to the localhost where you can view your web page.

Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

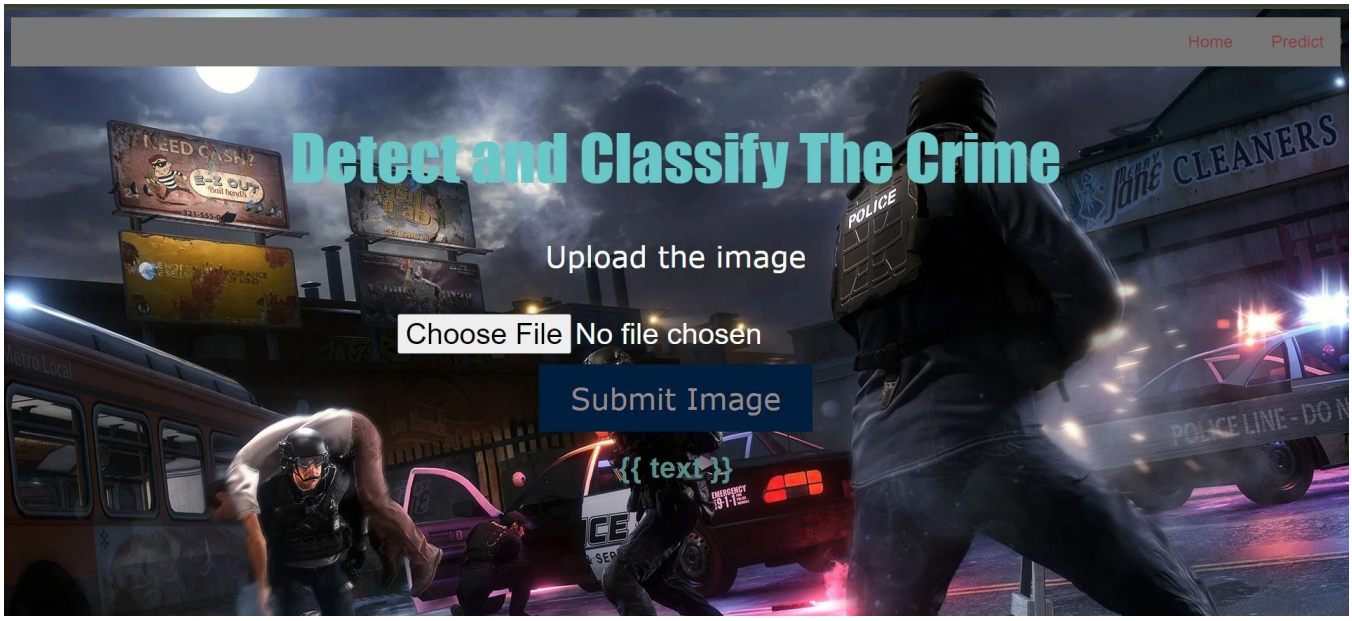
```
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
2023-11-06 23:52:29.324985: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimize
critical operations.
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX AVX2 FMA, in other operations, rebuild Tensor
 * Debugger is active!
 * Debugger PIN: 936-275-016
```

Now, Go the web browser and write the localhost url (http://127.0.0.1:5000) to get the below result.

Index page:

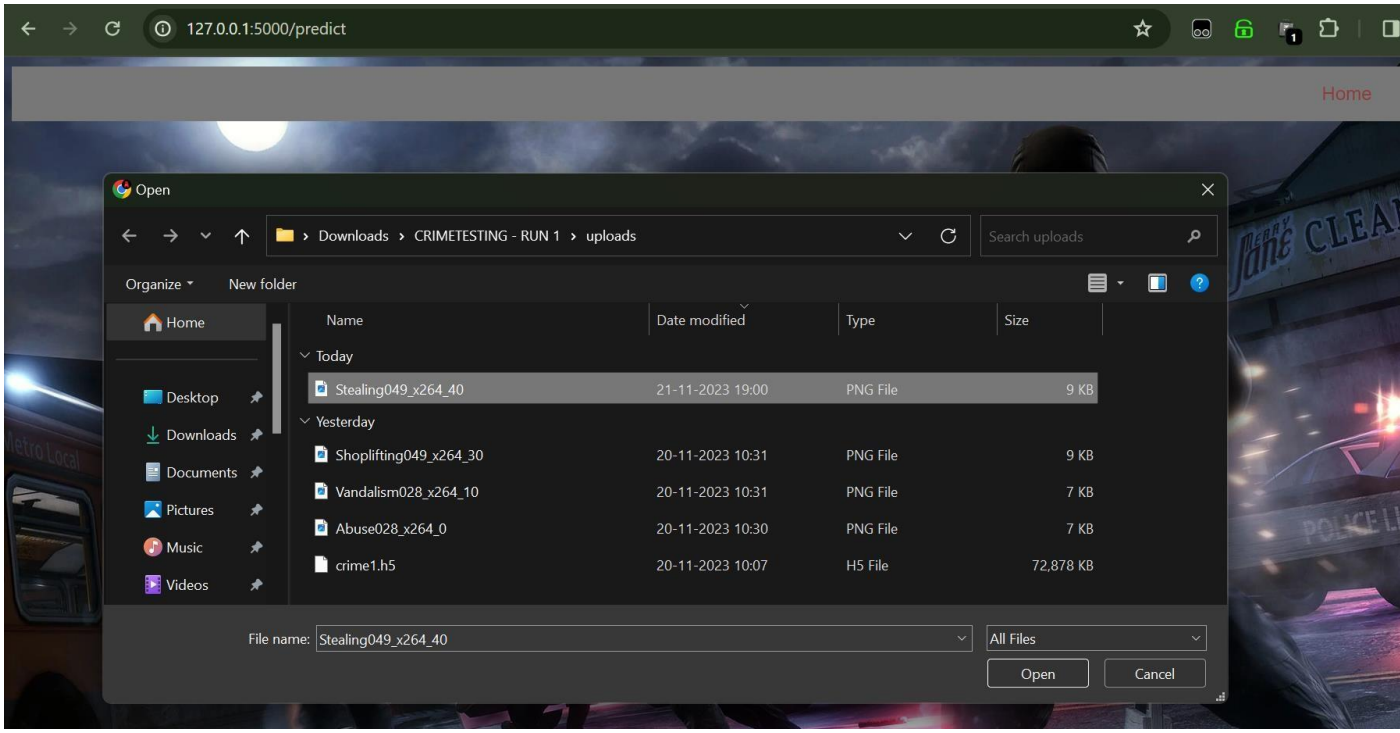


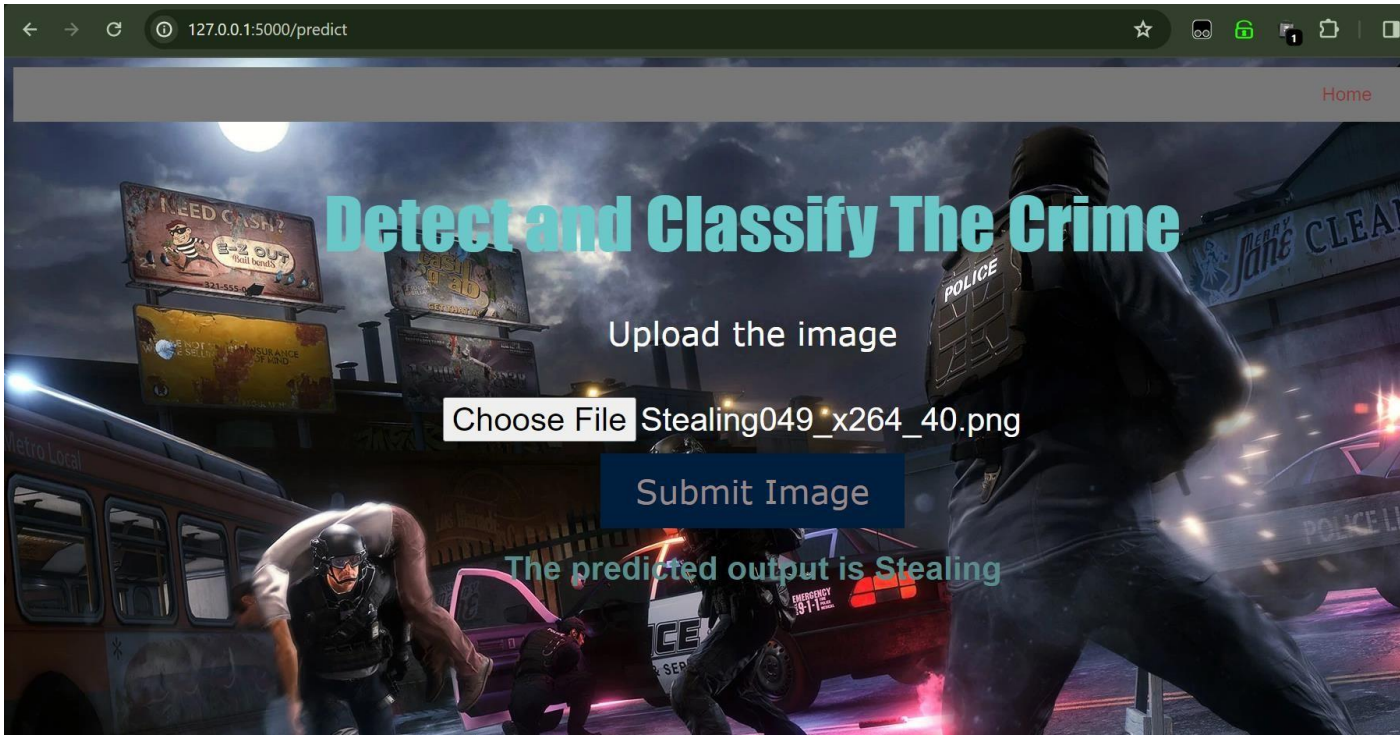
Prediction page:



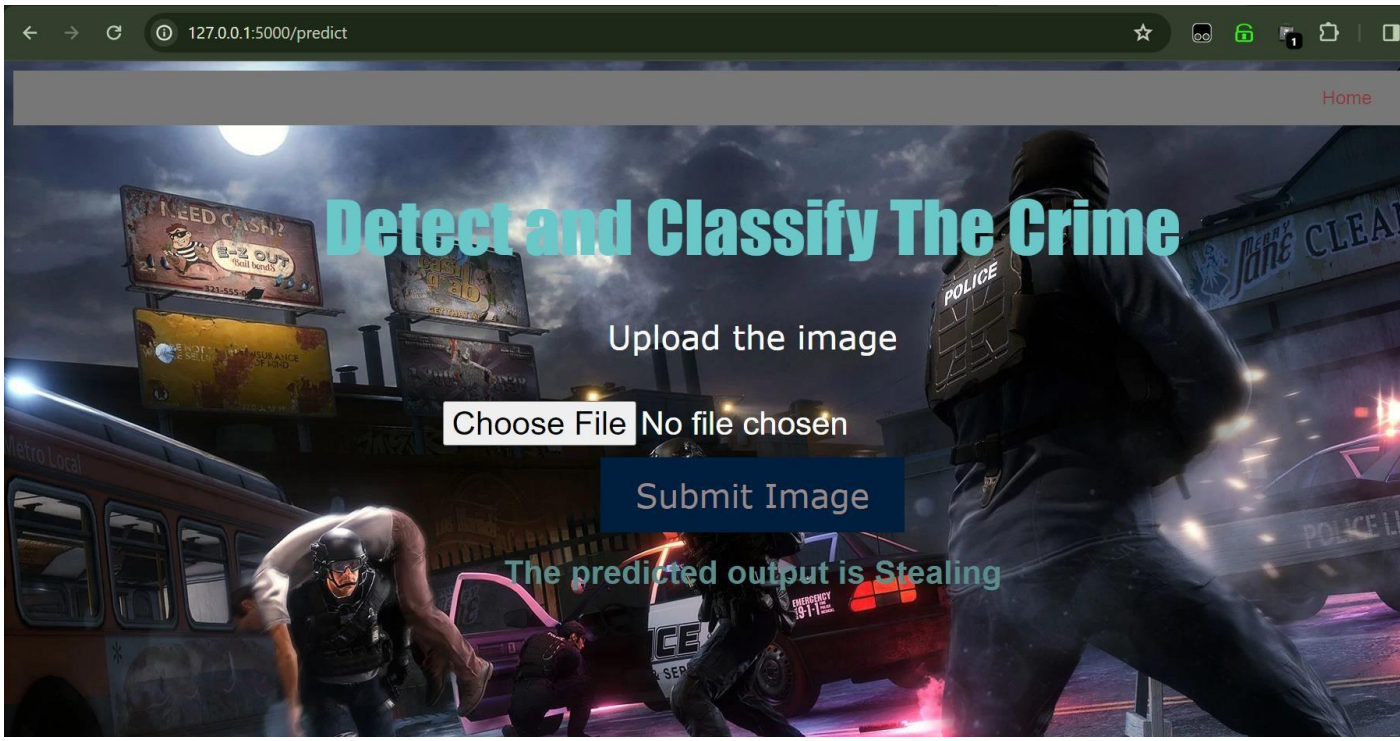
Predicting with various input images

Input 1:

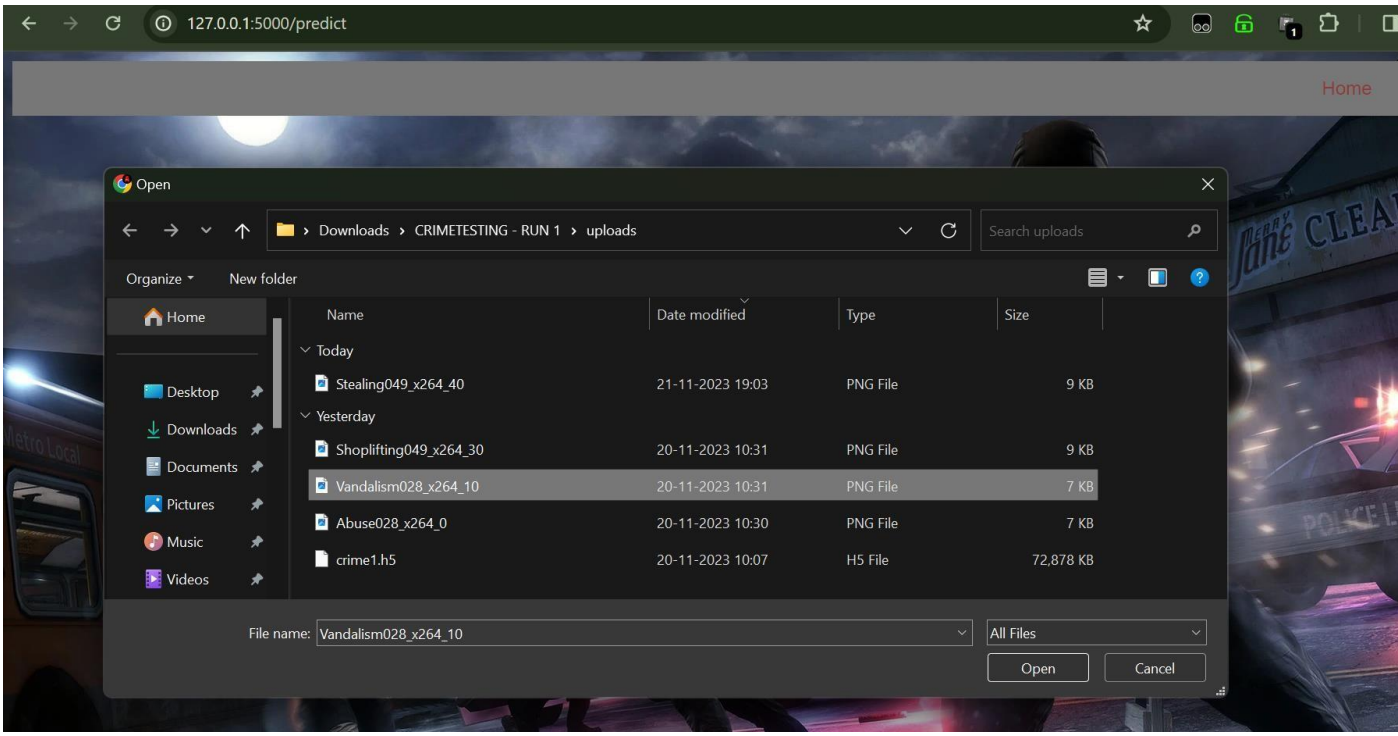


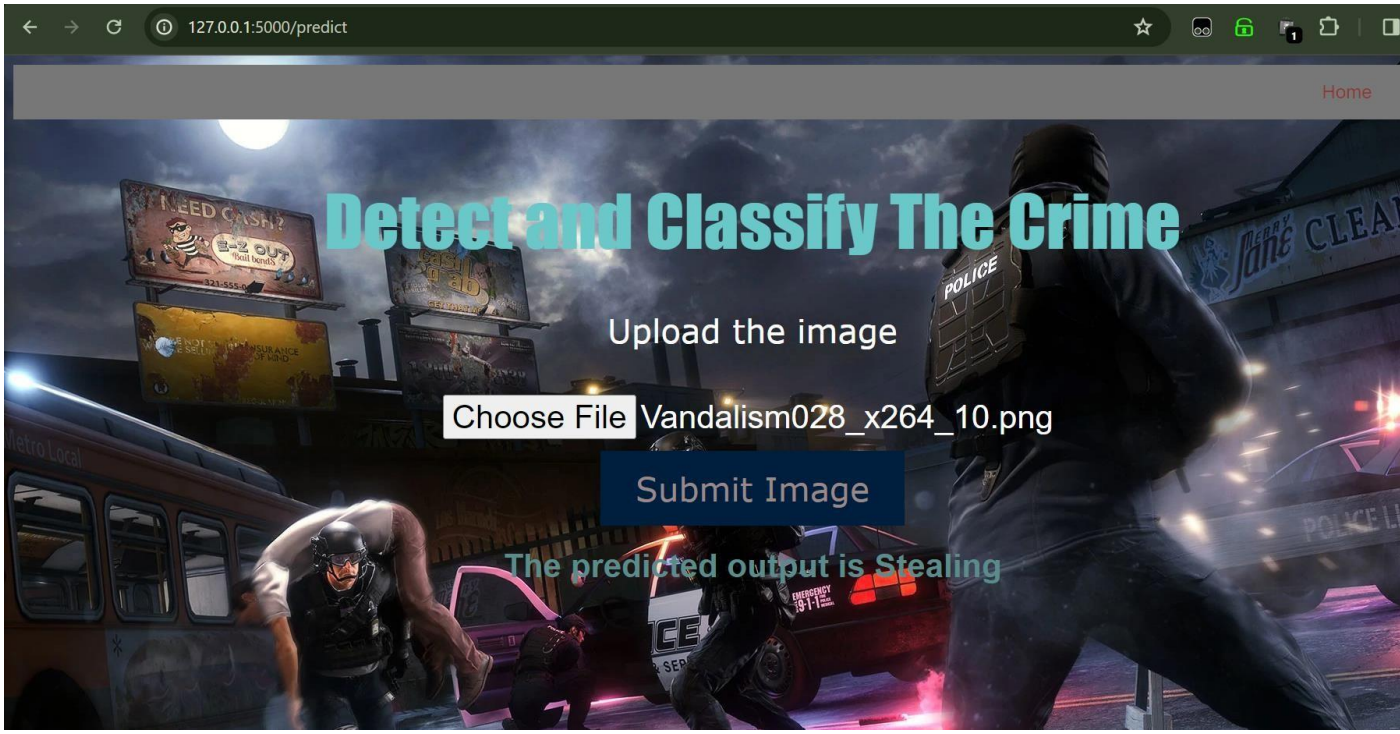

127.0.0.1:5000/predict
Home
Detect and Classify The Crime
Upload the image
Choose File
Stealing049_x264_40.png
Submit Image
The predicted output is Stealing

Output 1:

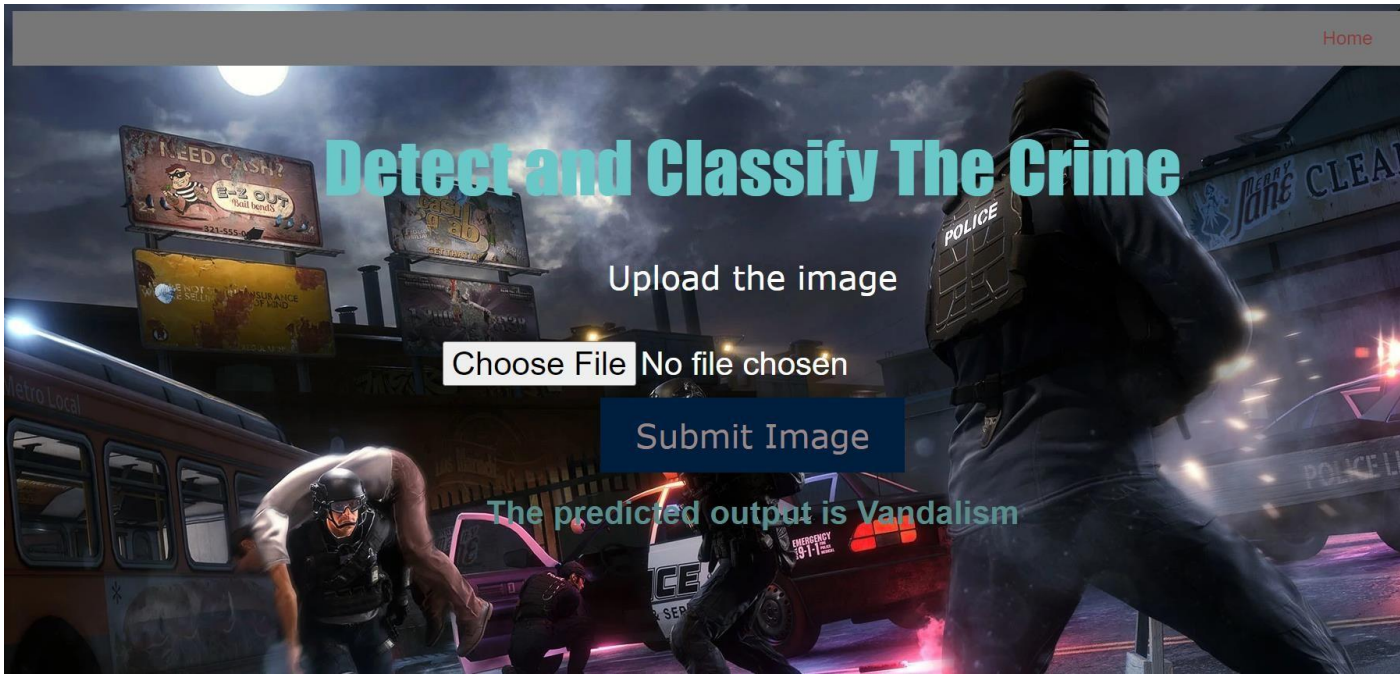

127.0.0.1:5000/predict
Home
Detect and Classify The Crime
Upload the image
Choose File
No file chosen
Submit Image
The predicted output is Stealing

Input 2:


127.0.0.1:5000/predict
Home
Open
Downloads › CRIMETESTING - RUN 1 › uploads
Search uploads
Organize
New folder
Home
Desktop
Downloads
Documents
Pictures
Music
Videos
Name
Date modified
Type
Size
Today
Stealing049_x264_40
21-11-2023 19:03
PNG File
9 KB
Yesterday
Shoplifting049_x264_30
20-11-2023 10:31
PNG File
9 KB
Vandalism028_x264_10
20-11-2023 10:31
PNG File
7 KB
Abuse028_x264_0
20-11-2023 10:30
PNG File
7 KB
crime1.h5
20-11-2023 10:07
H5 File
72,878 KB
File name:
Vandalism028_x264_10
All Files
Open
Cancel



Output 2:



10) Advantages & Disadvantages

Crime classification using Artificial Intelligence and Machine Learning (AIML) comes with several advantages and disadvantages.

Advantages:

- Efficiency: Crime classification using AIML allows for swift and automated processing of large datasets, enabling law enforcement to categorize and analyze crimes efficiently.
- Pattern Recognition:AIML algorithms excel at identifying patterns within data. In crime classification, this capability enhances the system's ability to recognize subtle correlations and trends in criminal activities.
- Predictive Analysis:Machine Learning models can predict potential criminal activities based on historical data, assisting law enforcement in proactive measures and resource allocation.
- Real-time Monitoring:AIML systems can continuously monitor live data feeds, providing real-time insights into ongoing criminal incidents and enabling rapid response.
- Improved Accuracy:The use of advanced algorithms enhances the accuracy of crime classification, reducing the likelihood of misclassifications and false identifications.

Disadvantages:

- Data Bias:AIML models heavily rely on historical data, and if the data used for training contains biases, the system may perpetuate or even exacerbate existing biases in crime classification.
- Complexity and Interpretability:Some AIML algorithms, particularly deep learning models, are complex and challenging to interpret. Understanding the decision-making process of these models can be intricate, impacting trust and transparency.
- Over-reliance on Technology:A complete reliance on AIML for crime classification may lead to a reduction in human involvement and oversight, potentially overlooking contextual nuances that a human analyst might consider.
- Privacy Concerns:The use of AIML in crime classification often involves the analysis of sensitive personal data. Balancing the need for effective crime prevention with individual privacy rights is a crucial challenge.

11) Conclusion

While AIML brings significant advancements to crime classification, careful consideration of its limitations, ethical implications, and potential biases is essential for responsible and effective implementation.

12) Future Scope

The future scope of crime classification using Artificial Intelligence and Machine Learning (AIML) holds promising developments and advancements. Here are key aspects of the potential future scope:

Enhanced Predictive Analytics:

Future systems will likely incorporate more sophisticated machine learning algorithms, improving the accuracy and efficiency of crime prediction. Advanced predictive analytics will enable law enforcement to anticipate criminal activities with higher precision.

Integration of Multimodal Data:

The integration of various data sources, including video surveillance, social media, and sensor data, will enhance the holistic understanding of criminal patterns. AIML models will evolve to analyze multimodal data for comprehensive crime classification.

Real-time Crime Monitoring:

Future systems may focus on real-time crime monitoring, allowing law enforcement agencies to respond swiftly to unfolding events. Continuous data analysis and instant alerts will be crucial for proactive crime prevention.

Explainable AI for Transparency:

Addressing the challenge of interpretability, future AIML systems in crime classification may incorporate explainable AI techniques. This will ensure transparency in decision-making processes, enhancing user trust and facilitating legal and ethical considerations.

Continuous Model Learning:

To adapt to evolving criminal tactics, future AIML models may employ continuous learning mechanisms. These models will dynamically update based on new data, improving their resilience against emerging patterns and trends.

Privacy-Preserving Techniques:

With an increasing emphasis on privacy concerns, the future of crime classification using AIML may involve the development of techniques that balance the need for effective law enforcement with the protection of individual privacy rights.

Global Collaboration and Standardization:

The future may witness increased collaboration among countries and organizations to standardize crime data formats and AIML models. This global effort could lead to more effective cross-border crime detection and prevention.

In conclusion, the future scope of crime classification using AIML is characterized by advancements in predictive capabilities, integration of diverse data sources, real-time monitoring, and a focus on transparency and privacy. These developments aim to create more robust and ethically sound systems for crime prevention and law enforcement.

13)Appendix

Source code

https://drive.google.com/drive/folders/1_LZ02_4ebrYA-aiHHgPiqYXq43XwKI3x?usp=sharing

Gitub and Project demo link

VIDEO LINK : https://youtu.be/YjDUlRqUpOg

GITHUB (SMARTINTERNZ) : https://github.com/smartinternz02/SI-GuidedProject-612679-1699962006

GITHUB(PERSONAL REPO): https://github.com/byreddyakhileswarreddy/CRIME-VISION-ADVANCED-CRIME-DETECTION/tree/main