ROAD HAZARD PRONITY CLASSIFICATION AND MISHAP DETECTION USING ARTIFICIAL INTELLIGENCE

TEAM MEMBERS:

ADITYA NARAYAN(19BCE2172)

CONTENTS

1. INTRODUCTION

Traditional traffic systems are designed only to monitor traffic but it does not provide any solution to decrease the fatal accidental rate which occurs due to lack of medical aid in real time. Consider a scenario where an accident occurred but no one was there to report this accident, the victim is critical. We cannot root out accidents totally but we can improve in providing post-crash care by detecting the accident as quickly as possible. There are lots of sensors available in the market as well but that requires installation in vehicles. The sensors will trigger the system that will alert nearby medical assistance or an emergency contact number. But what if the accident happened for a vehicle which is not equipped with such a sensor-based system. We need an advanced Artificial intelligence-based surveillance system which can detect occurrences of accidents depending on different accident-prone regions. Our team will be taking into account various parameters such as Accident Severity, Pothole Severity, Weather Conditions etc ,and predictive techniques would be applied to label the given area as High, Medium and Low based on its Accident Severity. We would try to integrate the proposed system with various CCTV cameras and we would try to configure the frame rate according to the AccidentSeverity Label assigned to that region in section-1. So, if the area falls under More High severity region, then the live video frames captured will have shorter time duration between them. Moreover, after all such configuration ,the system will be able to detect accidents if any occurs.

2. HARDWARE/SOFTWARE REQUIREMENTS:

Hardware: System with minimum 8 GB RAM will suffice

Software: Jupyter Notebook/Google collab ,Kaggle software,Required Python Libraries

3. EXISTING SYSTEM/APPROACH/METHOD

Traditional traffic systems are solely meant to monitor traffic; they do not offer any solutions for lowering the fatal accident rate caused by a lack of medical assistance available in real time. Consider the following scenario: an accident occurs, but no one is present to report it; the victim is in grave danger. We won't be able to completely eliminate accidents, but we can improve post-crash care by recognising them as soon as feasible. There are a plethora of sensors on the market, but they all require vehicle installation. The sensors will activate a system that will notify nearby medical help or a phone number in case of an emergency.

3.1 DRAWBACK OF EXISTING METHOD

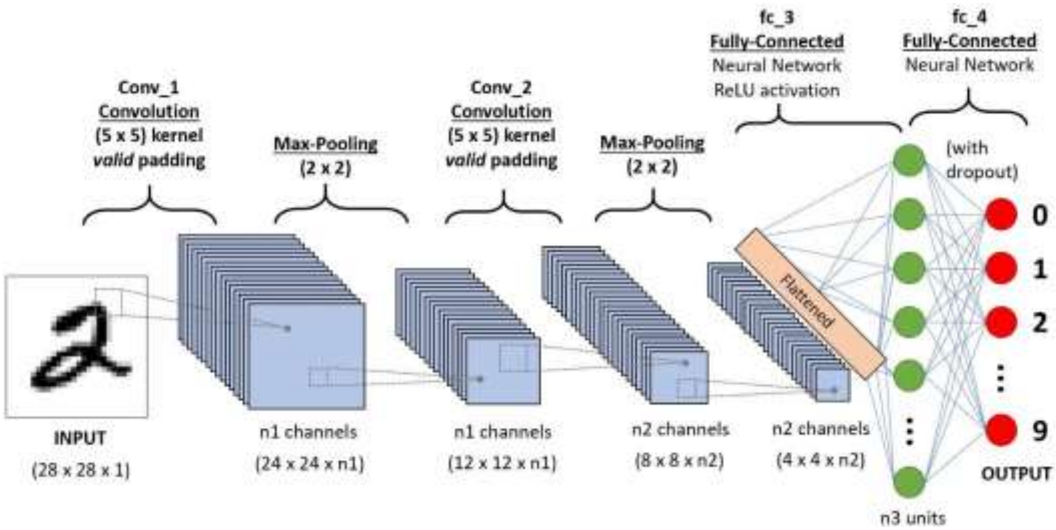
Consider a traditionally based vehicular device which faced an accident case.In that scenario no sense of information or alert would be sent to the required authorities.Thus a sense of technology was required which would solve the above mentioned issue.

4.4 PROPOSED MODEL

The project is divided into two parts:

Part 1:

In part-1 we have predicted an accident or no accident. We have used a dataset in which we have images which contain accidents and another set which contain images of no accident. We have used deep learning techniques for predictions. The images are converted into grayscale to remove chromium effects and noises. CNN is used for predictions. We have used the fast.ai library. Fastai library's goal is to make the training of deep neural networks as easy as possible, and, at the same time, make it fast and accurate using modern best practices. Using the fast.ai library we have made the implementation easy for us.



It’s based on the research into deep learning best practices under undertaken at fast.ai, a research institute founded and led by Jeremy Howard and Rachel Thomas in San Francisco.

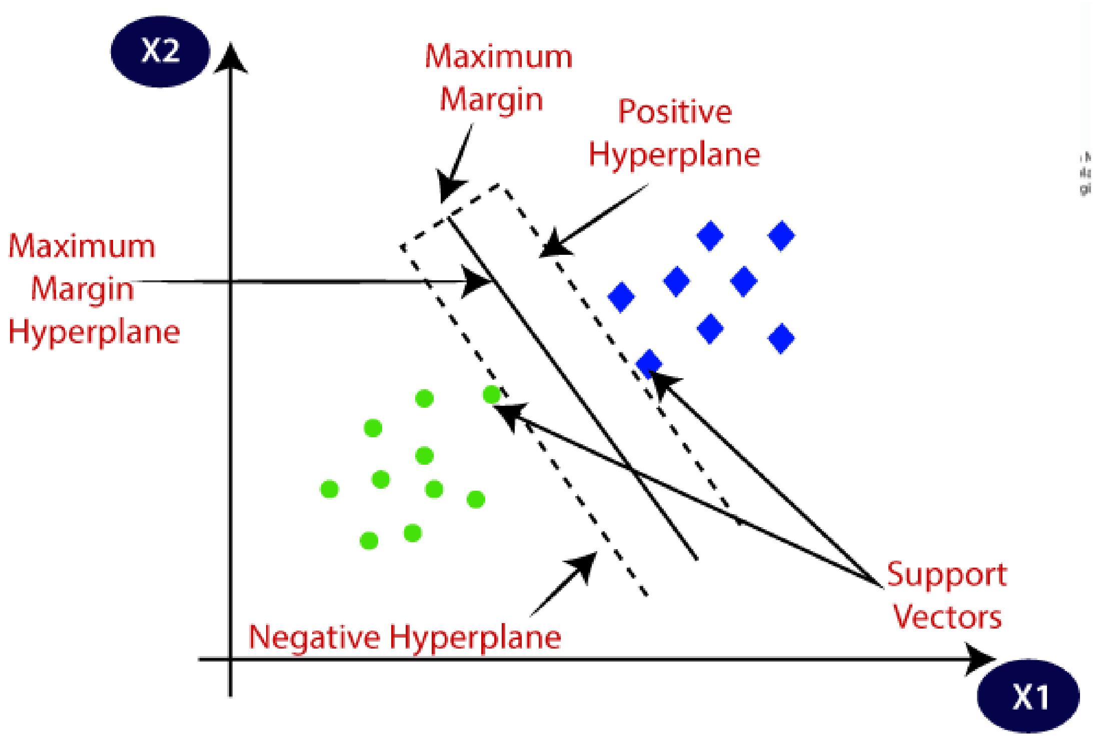
Part 2:

In part-2 we have predicted the severity of the accident. For this we have used the Support Vector Machine algorithm. Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

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.

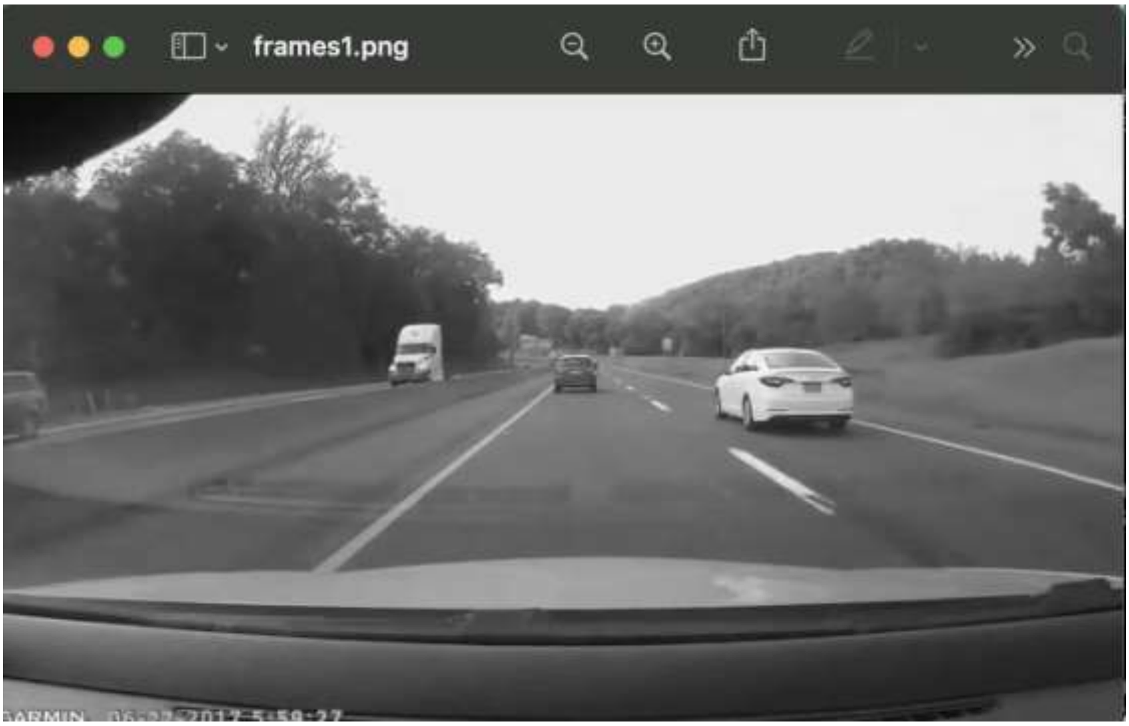
SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called support vectors, and hence the algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane.


X2
Maximum Margin
Positive Hyperplane
Maximum Margin Hyperplane
Support Vectors
Negative Hyperplane
X1

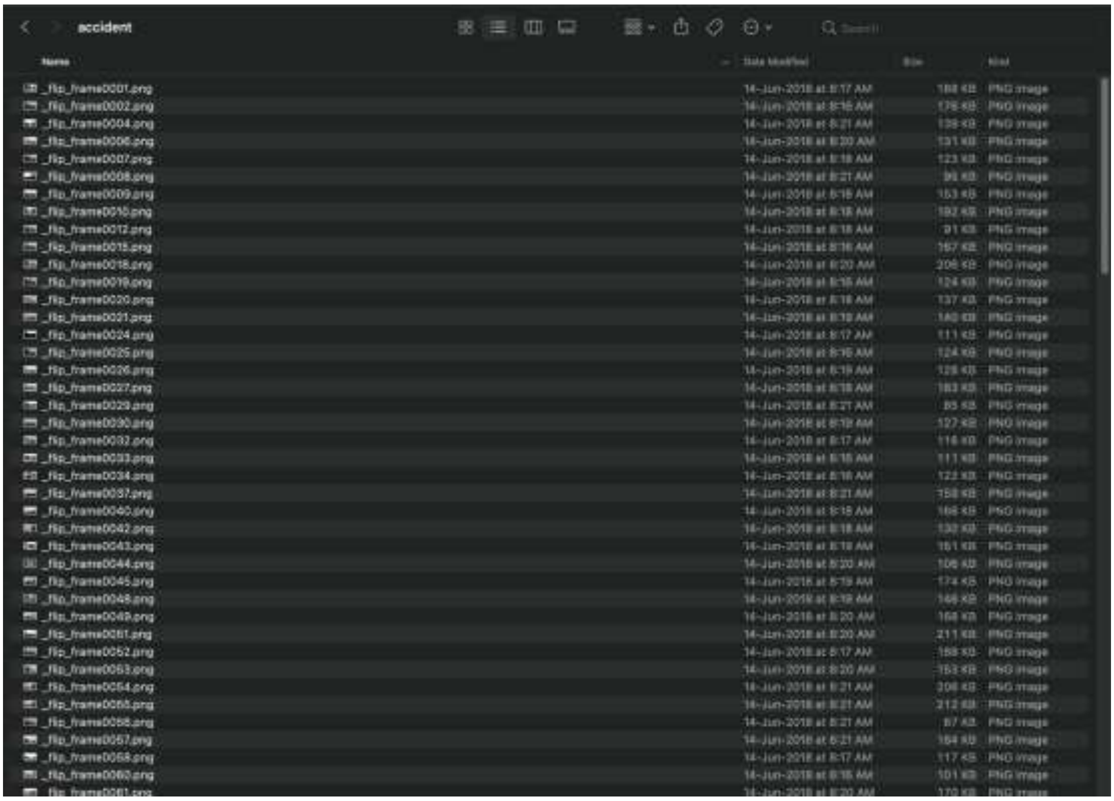
4.1 IMPLEMENTATION

Dataset for accident images:

Dataset for non-accident images:


frames1.png

‹›



File for predicting Accident or No-Accident

Import the fastai libraries

```python
# Put these at the top of every notebook, to get automatic reloading and inline plotting
%reload_ext autoreload
%autoreload 2
%matplotlib inline
```

```python
# This file contains all the main external libs we'll use
from fastai.imports import *
from fastai.transforms import *
from fastai.conv_learner import *
from fastai.model import *
from fastai.dataset import *
from fastai.sgdr import *
from fastai.plots import *
import sys
```

2.2 Load model

```python
###   Add here the name of the model to test
model_name = "carCrash180614-1506_sz299_bs60_ep5"
```

Python

```python
arch=resnet34
sz = 299                                                          # Type same size used during model crea
batch_size = 60                                                   # Type same batch_size used during mode
PATH = "GDrive/My Drive/CarCrashDetection/Dataset/"
data = ImageClassifierData.from_paths(PATH, tfms=tfms_from_model(arch, sz), bs=batch_size)
learn = ConvLearner.pretrained(arch, data, precompute=True)


print ( "Loading model:", model_name,".h5\n")
learn.load(model_name)
print ( "Model loaded")
```

Python

```
Downloading: "https://download.pytorch.org/models/resnet34-333f7ec4.pth" to /root/.torch/models/resnet34-333f7ec4.pth
100%|██████████| 87306240/87306240 [00:04<00:00, 18944882.52it/s]

Loading model: carCrash180614-1506_sz299_bs60_ep5 .h5

Model loaded
```

2.3 Load image to test

```python
PATH = "GDrive/My Drive/CarCrashDetection/Dataset/train/no_accident/"
fn = "4frame60.png"

PATH = "GDrive/My Drive/CarCrashDetection/Dataset/train/accident/"
fn = "frame0020.png"

img = open_image(PATH+fn)

print(img.shape)
plt.imshow(img);
```

Python

```
(360, 640, 3)
```

›

2.4 Prediction Test

```python
trn_tfms, val_tfms = tfms_from_model(arch,sz)       # get transformations
im = val_tfms (img)

learn.precompute=False                              # We'll pass in a raw image, not activations
preds = learn.predict_array(im[None])
np.argmax(preds)                                    # preds are log probabilities of classes
```

```
1
```

3. Prediction Batch processing Test

›

3.1 Define test bench folder to check

```python
PATH = "GDrive/My Drive/CarCrashDetection/video_to_check/frames4-noaccident/"



filenames_to_check = os.listdir ("GDrive/My Drive/CarCrashDetection/video_to_check/frames4-noaccident/")
number_images = len(filenames_to_check)
print(f"Number of images to check: {number_images}")
```

```
Number of images to check: 44
```

›

3.2 Prediction

```python
count = 0
list = []
for i in range(number_images):
  fn = filenames_to_check[i]
  img = open_image(PATH+fn)
  trn_tfms, val_tfms = tfms_from_model(arch,sz)
  im = val_tfms (img)

  learn.precompute=False
  preds = learn.predict_array(im[None])
  p = np.argmax(preds)
  list.append(p)
  count += 1
  if (count%5):
    b = round((count*100 / number_images),0)
    sys.stdout.write('\r'+ str(b) + "%")

print ("")
print ("Imagenes procesadas :", count)
print ("Tentative number of no accidents :", list.count(1))
print ("Tentative number of accidents :", list.count(0))
```

```
100.0%
Imagenes procesadas : 44
```

```
Tentative number of no accidents : 44
Tentative number of accidents : 0
```

3.3 Prediction normalization

```python
# normalization setup
seg_lenght = 5      # Segment lenght to analize
steps = 4           # number of steps between length
positivos = 2       # number of '0's in the lenght
```

```python
list_normalized = []


for i in range(0, (len(list)-(steps+1)), steps):                    # iterates the prediction list according the steps
  norm = 0
  for j in range (seg_lenght):
    posicion = i+j
    norm = norm + list[posicion]

  if (norm>positivos):
    list_normalized.append(1)
  else:
    list_normalized.append(0)
```

```python
list_normalized = []


for i in range(0, (len(list)-(steps+1)), steps):                    # iterates the prediction list according the steps
  norm = 0
  for j in range (seg_lenght):
    posicion = i+j
    norm = norm + list[posicion]

  if (norm>positivos):
    list_normalized.append(1)
  else:
    list_normalized.append(0)
```

```python
print (list)
print (list_normalized)
```

```
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

3.4 result

```python
count = 0
accidente = 0
for i in range( len (list_normalized)):
  if (count == 3):
    accidente = 1
    break
  else:
    if (list_normalized[i] == 1):
      count == 0
    else:
      count +=1
if (accidente == 0):
  print ("No accident in video")
else:
  print ("Warning, accident")
```

```
No accident in video
```

File to predict severity of accident

Accident Severity Prediction

Importing necessary libraries

```python
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import r2_score, mean_squared_error
import matplotlib.pyplot as plt
```

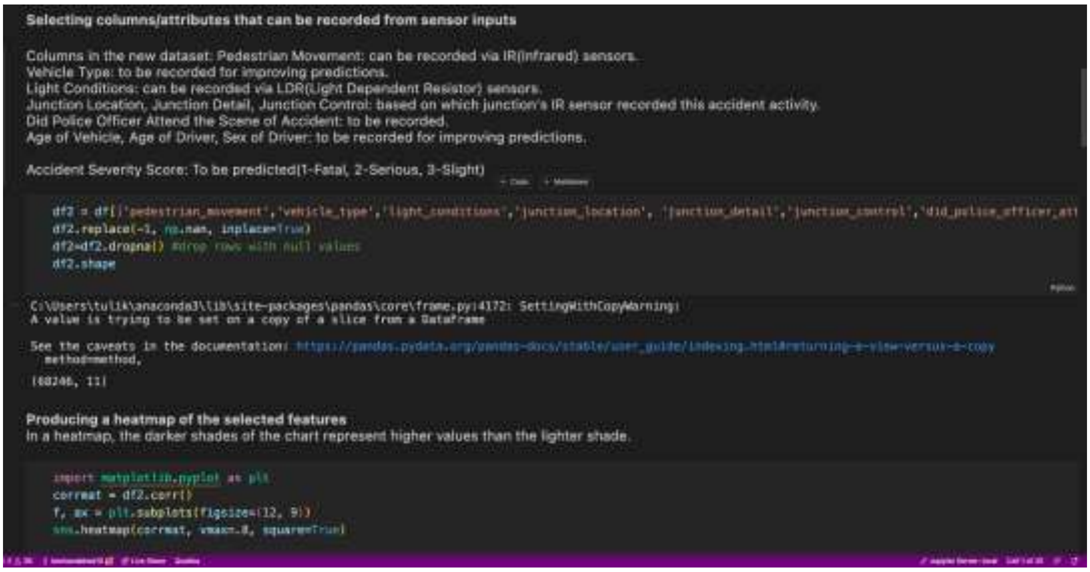
Importing the dataset and dropping rows with null values

```python
df=pd.read_csv("dataset.csv")
df=df.dropna()
df.head()
```

```
C:\Users\tulik\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3063: DtypeWarning: Columns (0,55) have mixed types.Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

	accident_index	vehicle_reference	vehicle_type	towing_and_articulation	vehicle_manoeuvre	vehicle_location-restricted_lane	junction_location	skidding_and_overturning	hit_object_in_carriage
2	201506E098766	2	9	0	18	0	8	0	
3	201506E098777	1	20	0	4	0	0	0	
5	201506E098780	2	1	0	9	0	5	0	
6	201506E098792	1	3	0	4	0	2	0	
8	201506E098804	1	9	0	14	0	1	0	

5 rows × 70 columns



Selecting columns/attributes that can be recorded from sensor inputs

Columns in the new dataset: Pedestrian Movement: can be recorded via IR(Infrared) sensors.
Vehicle Type: to be recorded for improving predictions.
Light Conditions: can be recorded via LDR(Light Dependent Resistor) sensors.
Junction Location, Junction Detail, Junction Control: based on which junction's IR sensor recorded this accident activity.
Did Police Officer Attend the Scene of Accident: to be recorded.
Age of Vehicle, Age of Driver, Sex of Driver: to be recorded for improving predictions.

Accident Severity Score: To be predicted(1-Fatal, 2-Serious, 3-Slight)

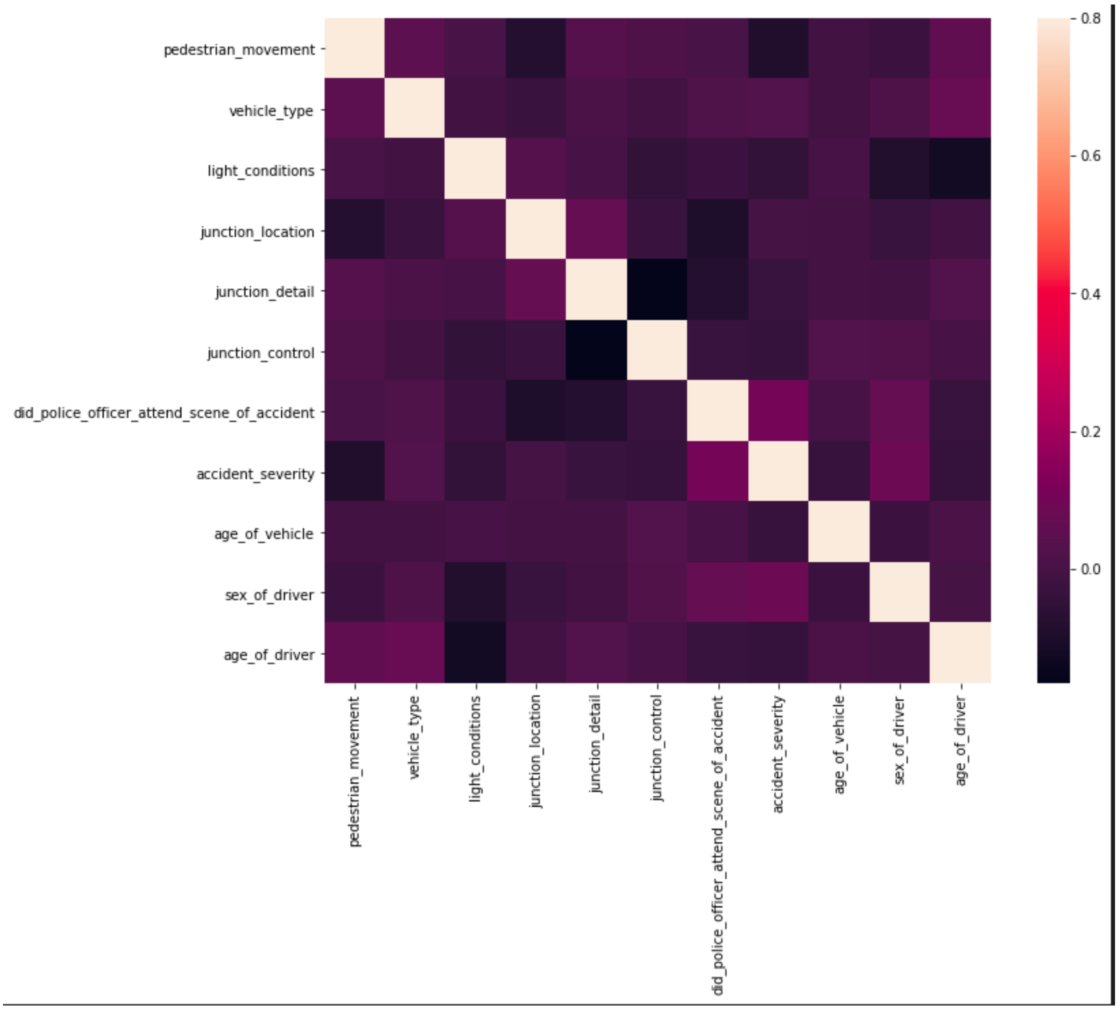
```
df2 = df[['pedestrian_movement','vehicle_type','light_conditions','junction_location', 'junction_detail','junction_control','did_police_officer_att
df2.replace(-1, np.nan, inplace=True)
df2=df2.dropna() #drop rows with null values
df2.shape
```

```
C:\Users\tulik\anaconda3\lib\site-packages\pandas\core\frame.py:4172: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  method=method,
(68246, 11)
```

Producing a heatmap of the selected features
In a heatmap, the darker shades of the chart represent higher values than the lighter shade.

```
import matplotlib.pyplot as plt
corrmat = df2.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, vmax=.8, square=True)
```


pedestrian_movement
vehicle_type
light_conditions
junction_location
junction_detail
junction_control
did_police_officer_attend_scene_of_accident
accident_severity
age_of_vehicle
sex_of_driver
age_of_driver
0.8
0.6
0.4
0.2
0.0

››

```
df2.head()
```

	pedestrian_movement	vehicle_type	light_conditions	junction_location	junction_detail	junction_control	did_police_officer_attend_scene_of_accident	accident_severity	age_of_
6	0.0	3.0	7	2.0	6.0	2.0	1.0	3	
8	2.0	9.0	1	1.0	3.0	2.0	1.0	3	
14	0.0	9.0	1	8.0	6.0	4.0	1.0	3	
25	0.0	9.0	1	8.0	3.0	4.0	1.0	3	
26	3.0	9.0	1	8.0	3.0	4.0	1.0	3	

```
df= df[:15000]
df2= df2[:15000] #keep 15000
```

```
Y = df2.accident_severity.values
Y1 = df.accident_severity.values
Y
```

```
array([3, 3, 3, ..., 3, 3, 3], dtype=int64)
```

```
cols = df2.shape[1]
X = df2.loc[:, df2.columns != 'accident_severity']
X1 = df.loc[:, df.columns != 'accident_severity']
X.columns
```

```
Index(['pedestrian_movement', 'vehicle_type', 'light_conditions',
       'junction_location', 'junction_detail', 'junction_control',
       'did_police_officer_attend_scene_of_accident', 'age_of_vehicle',
       'sex_of_driver', 'age_of_driver'],
      dtype='object')
```

```
print(X.shape)
print(X1.shape)
```

```
(15000, 10)
(15000, 69)
```

```
# Support Vector Machines
X_train, X_test,Y_train,Y_test = train_test_split(X, Y, test_size=0.33, random_state=99)
```

Fitting the prediction model and calculating accuracy

```
from sklearn.svm import SVC, LinearSVC

svc = SVC()
svc.fit(X_train, Y_train)
Y_pred = svc.predict(X_test)
acc_svc = round(svc.score(X_test, Y_test) * 100, 2)
print("Accuracy using SVM=", acc_svc)
```

```
Accuracy using SVM= 92.06
```

Test Input

The test inputs include the following:

Pedestrian Movement = 1.0 = Crossing from driver's nearside
Vehicle Type = 2.0 = Motorcycle 50cc and under
Light Conditions = 4 = Darkness - lights lit
Junction Location = 1.0 = Approaching junction or waiting/parked at junction approach
Junction Detail = 6.0 = Crossroads

5. RESULT:

For the depiction of accident several datasets(both of accident and non-accident) were collected.They were processed through a given set of mechanisms and the predictability of an occurrence of an accident and its severity was achieved.The accuracy attained during the attainability of the given set of prediction was satisfactory.

6. CONCLUSION:

Thus, for each type of Accident-Prone areas (High/Medium/Low) successfully we are able to adjust the frames-per-second to be processed in our system and they are divided into various clusters and we are also able to detect accidents in the video and adjusting the frame according to the area type. There are lots of sensors available in the market as well but that requires installation in vehicles.

7. REFERENCES:

1. B. Alexe, T. Deselaers, V. Ferrari, “Measuring the objectness of image windows”, TPAMI, 2012. [2] Guzel, MS, “Versatile Vehicle Tracking and Counting Application”,KaraElmas Science and Eng Journal,7(2),622- 626,2017

2. Liang-Chien Liu, Chiung-Yao Fang, Sei-Wang Chen, A Novel Distance Estimation Method Leading a Forward Collision Avoidance Assist System for Vehicles on Highways, IEEE Transactions on Intelligent Transportation Systems (Volume: 18, Issue: 4, April 2017)

3. Linder, Astrid & Avery, Matthew. (2001). Change of velocity and pulse characteristics in rear impacts: real world and vehicle tests data.

Gabauer DJ, Gabler HC. Comparison of delta-v and occupant impact velocity crash severity metrics using

event data recorders. Annu Proc Assoc Adv Automot Med. 2006

4.Wang, K., Li, Z., Yao, Q., Huang, W. and Wang, F.Y., 2007, December. An automated vehicle counting system for traffic surveillance. In 2007 IEEE International Conference on Vehicular Electronics and Safety (pp. 1-6). IEEE.

5.Salvi, G., 2014, March. An automated nighttime vehicle counting and detection system for traffic surveillance. In 2014 International Conference on Computational Science and Computational Intelligence (Vol. 1, pp. 131-136). IEEE.