```
                num = 0
                i = i + 1
        else:
            m_exp[i] = str(item)
            m_exp[i] = m_exp[i].replace("10","/")
            m_exp[i] = m_exp[i].replace("11","+")
            m_exp[i] = m_exp[i].replace("12","-")
            m_exp[i] = m_exp[i].replace("13","*")

            i = i + 1


    'joining the list of strings to create the mathematical expression'
    separator = ' '
    m_exp_str = separator.join(m_exp)

    return (m_exp_str)

'creating the mathematical expression'
m_exp_str = math_expression_generator(elements_pred)
print(m_exp_str)
'calculating the mathematical expression using eval()'
while True:
    try:
        answer = eval(m_exp_str)    #evaluating the answer
        answer = round(answer, 2)
        equation  = m_exp_str + " = " + str(answer)
        print(equation)   #printing the equation
        break

    except SyntaxError:
        print("Invalid predicted expression!!")
        print("Following is the predicted expression:")
        print(m_exp_str)
        break
```

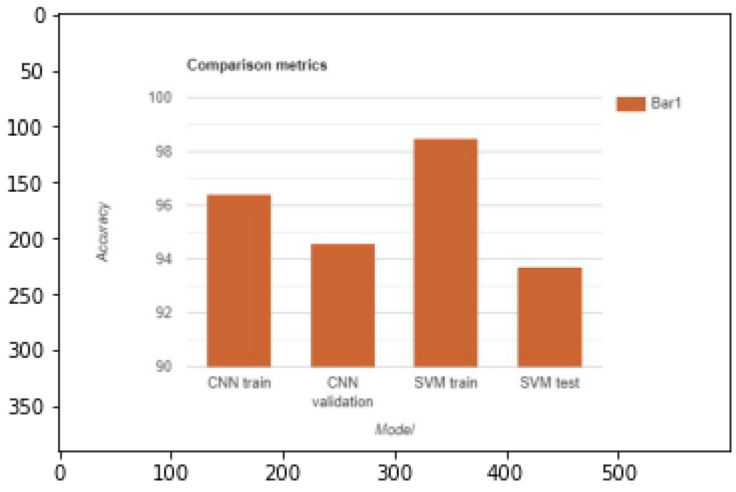
```
98 / 76 + 54 + 32 - 10
98 / 76 + 54 + 32 - 10 = 77.29
```

In [ ]:

```
img = cv2.imread("img1.jpeg")
plt.imshow(img)
```

Out[ ]:

```
<matplotlib.image.AxesImage at 0x246c5b13160>
```



NUMBER PLATE RECOGNITION

```
import matplotlib.pyplot as plt
```

In [ ]:

```
import numpy as np
import cv2
import tensorflow as tf
from sklearn.metrics import f1_score
from tensorflow.keras import optimizers
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense, Flatten, MaxPooling2D, Dropout, Conv2D
```

DATASET

In [ ]:

```
import tensorflow.keras.backend as K
train_datagen = ImageDataGenerator(rescale=1./255, width_shift_range=0.1, height_sl
path = '/content/drive/MyDrive/data'
train_generator = train_datagen.flow_from_directory(
        path+'/train',  # this is the target directory
        target_size=(28,28),  # all images will be resized to 28x28
        batch_size=1,
        class_mode='sparse')

validation_generator = train_datagen.flow_from_directory(
        path+'/val',  # this is the target directory
        target_size=(28,28),  # all images will be resized to 28x28 batch_size=1,
        class_mode='sparse')
```

```
Found 864 images belonging to 36 classes.
Found 216 images belonging to 36 classes.
```

In [ ]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [ ]:

```
def f1score(y, y_pred):
  return f1_score(y, tf.math.argmax(y_pred, axis=1), average='micro')

def custom_f1score(y, y_pred):
  return tf.py_function(f1score, (y, y_pred), tf.double)
```

MODEL

In [ ]:

```
K.clear_session()
model = Sequential()
model.add(Conv2D(16, (22,22), input_shape=(28, 28, 3), activation='relu', padding=
model.add(Conv2D(32, (16,16), input_shape=(28, 28, 3), activation='relu', padding=
model.add(Conv2D(64, (8,8), input_shape=(28, 28, 3), activation='relu', padding='s
model.add(Conv2D(64, (4,4), input_shape=(28, 28, 3), activation='relu', padding='s
model.add(MaxPooling2D(pool_size=(4, 4)))
model.add(Dropout(0.4))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(36, activation='softmax'))

model.compile(loss='sparse_categorical_crossentropy', optimizer=optimizers.Adam(lr=
```

```
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/adam.py:105: UserWarnin
g: The `lr` argument is deprecated, use `learning_rate` instead.
  super(Adam, self).__init__(name, **kwargs)
```

In [ ]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 16)        23248     
                                                                 
 conv2d_1 (Conv2D)           (None, 28, 28, 32)        131104    
                                                                 
 conv2d_2 (Conv2D)           (None, 28, 28, 64)        131136    
                                                                 
 conv2d_3 (Conv2D)           (None, 28, 28, 64)        65600     
                                                                 
 max_pooling2d (MaxPooling2D  (None, 7, 7, 64)         0         
 )                                                               
                                                                 
 dropout (Dropout)           (None, 7, 7, 64)          0         
                                                                 
 flatten (Flatten)           (None, 3136)              0         
                                                                 
 dense (Dense)               (None, 128)               401536    
                                                                 
 dense_1 (Dense)             (None, 36)                4644      
                                                                 
=================================================================
Total params: 757,268
Trainable params: 757,268
Non-trainable params: 0
_________________________________________________________________
```

TRAINING

In [ ]:

```
class stop_training_callback(tf.keras.callbacks.Callback):
  def on_epoch_end(self, epoch, logs={}):
    if(logs.get('val_custom_f1score') > 0.99):
      self.model.stop_training = True
```

In [ ]:

```
batch_size = 1
callbacks = [stop_training_callback()]
model.fit_generator(
      train_generator,
      steps_per_epoch = train_generator.samples // batch_size,
      validation_data = validation_generator,
      epochs = 15, verbose=1, callbacks=callbacks)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: UserWarning: `Mode
l.fit_generator` is deprecated and will be removed in a future version. Please use
`Model.fit`, which supports generators.
  import sys
```

```
Epoch 1/15
864/864 [==============================] - 273s 314ms/step - loss: 3.0760 - custom
_f1score: 0.1551 - val_loss: 2.1411 - val_custom_f1score: 0.3795
Epoch 2/15
864/864 [==============================] - 81s 93ms/step - loss: 1.3598 - custom_f
1score: 0.5984 - val_loss: 0.7278 - val_custom_f1score: 0.8021
Epoch 3/15
864/864 [==============================] - 81s 94ms/step - loss: 0.5892 - custom_f
1score: 0.8032 - val_loss: 0.5164 - val_custom_f1score: 0.8661
Epoch 4/15
864/864 [==============================] - 80s 93ms/step - loss: 0.4503 - custom_f
1score: 0.8715 - val_loss: 0.2745 - val_custom_f1score: 0.8988
Epoch 5/15
864/864 [==============================] - 81s 93ms/step - loss: 0.3128 - custom_f
1score: 0.8958 - val_loss: 0.1785 - val_custom_f1score: 0.9539
Epoch 6/15
864/864 [==============================] - 78s 90ms/step - loss: 0.2811 - custom_f
1score: 0.9086 - val_loss: 0.2859 - val_custom_f1score: 0.9033
Epoch 7/15
864/864 [==============================] - 79s 91ms/step - loss: 0.2192 - custom_f
1score: 0.9398 - val_loss: 0.1434 - val_custom_f1score: 0.9568
Epoch 8/15
864/864 [==============================] - 82s 94ms/step - loss: 0.1842 - custom_f
1score: 0.9421 - val_loss: 0.0975 - val_custom_f1score: 0.9777
Epoch 9/15
864/864 [==============================] - 80s 93ms/step - loss: 0.1746 - custom_f
1score: 0.9421 - val_loss: 0.1967 - val_custom_f1score: 0.9479
Epoch 10/15
864/864 [==============================] - 79s 91ms/step - loss: 0.1458 - custom_f
1score: 0.9572 - val_loss: 0.1218 - val_custom_f1score: 0.9673
Epoch 11/15
864/864 [==============================] - 80s 92ms/step - loss: 0.1263 - custom_f
1score: 0.9549 - val_loss: 0.0687 - val_custom_f1score: 0.9717
Epoch 12/15
864/864 [==============================] - 80s 93ms/step - loss: 0.1200 - custom_f
1score: 0.9653 - val_loss: 0.2610 - val_custom_f1score: 0.9315
Epoch 13/15
864/864 [==============================] - 79s 92ms/step - loss: 0.1253 - custom_f
1score: 0.9549 - val_loss: 0.0711 - val_custom_f1score: 0.9613
Epoch 14/15
864/864 [==============================] - 80s 92ms/step - loss: 0.0699 - custom_f
1score: 0.9734 - val_loss: 0.1193 - val_custom_f1score: 0.9464
Epoch 15/15
864/864 [==============================] - 80s 92ms/step - loss: 0.1424 - custom_f
1score: 0.9549 - val_loss: 0.0535 - val_custom_f1score: 0.9673
```

Out[ ]:

```
<keras.callbacks.History at 0x7f26366c2210>
```

ROI AND BBOX FOR NUMBER PLATE

In [ ]:

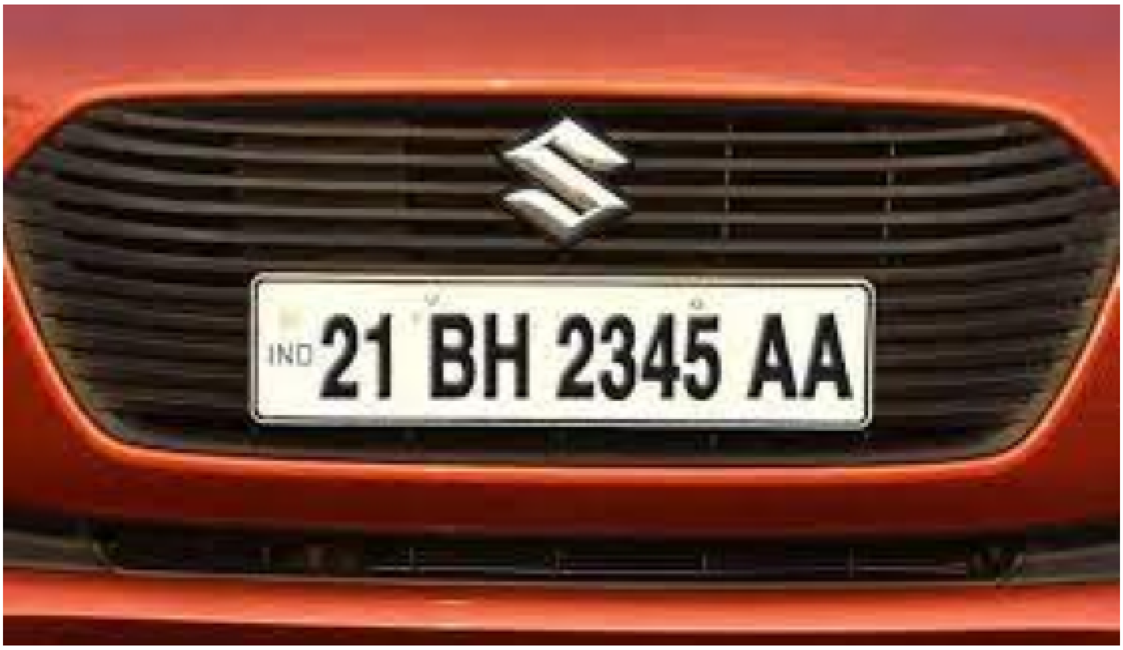
```
plate_cascade = cv2.CascadeClassifier('/content/drive/MyDrive/data/indian_license_
```

In [ ]:

```
def detect_plate(img, text=''):
    plate_img = img.copy()
    roi = img.copy()
    plate_rect = plate_cascade.detectMultiScale(plate_img, scaleFactor = 1.2, minN
    for (x,y,w,h) in plate_rect:
        roi_ = roi[y:y+h, x:x+w, :] # extract ROI
        plate = roi[y:y+h, x:x+w, :]
        cv2.rectangle(plate_img, (x+2,y), (x+w-3, y+h-5), (51,181,155), 3) # bbox
    if text!='':
        plate_img = cv2.putText(plate_img, text, (x-w//2,y-h//2),
                                cv2.FONT_HERSHEY_COMPLEX_SMALL , 0.5, (51,181,155)
    return plate_img, plate
img = cv2.imread('/content/drive/MyDrive/data/car.jpg')
img2 = cv2.imread('/content/drive/MyDrive/data/car3.jpg')
```

In [ ]:
```
def display(img_, title=''):
    img = cv2.cvtColor(img_, cv2.COLOR_BGR2RGB)
    fig = plt.figure(figsize=(10,6))
    ax = plt.subplot(111)
    ax.imshow(img)
    plt.axis('off')
    plt.title(title)
    plt.show()

display(img)
```

In [ ]:
```
display(img2)
```



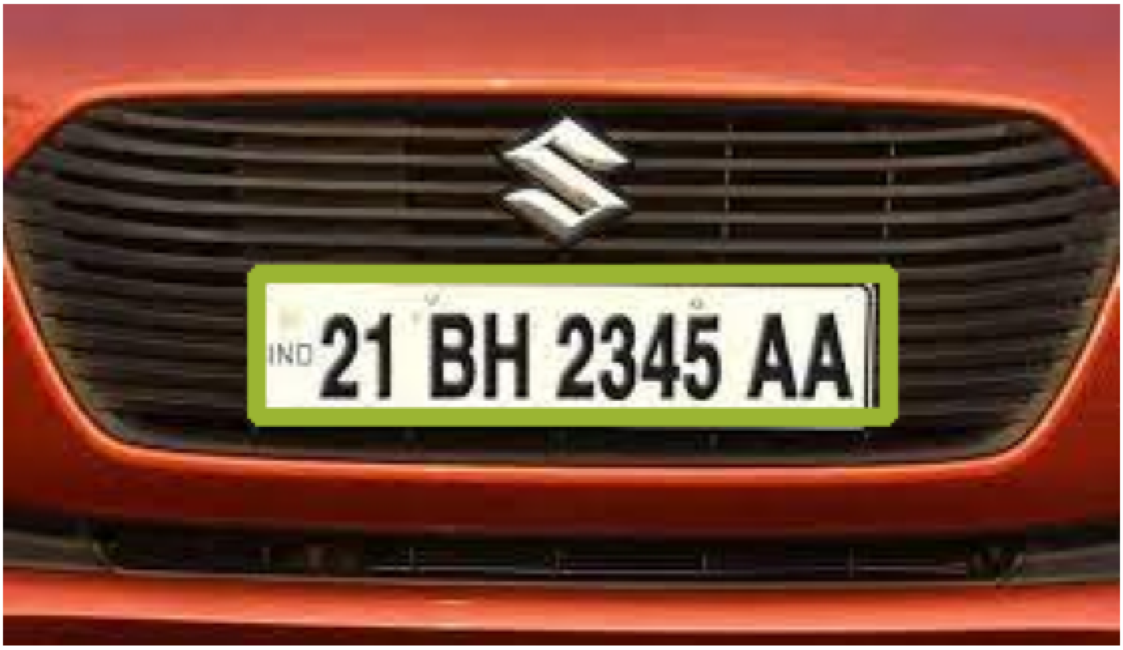
In [ ]:
```
output_img, plate1 = detect_plate(img)
output_img2, plate2 = detect_plate(img2)
```

In [ ]:
```
display(output_img, 'detected license plate in the input image')
```

In [ ]:
```
display(plate1, 'extracted license plate from the image')
```

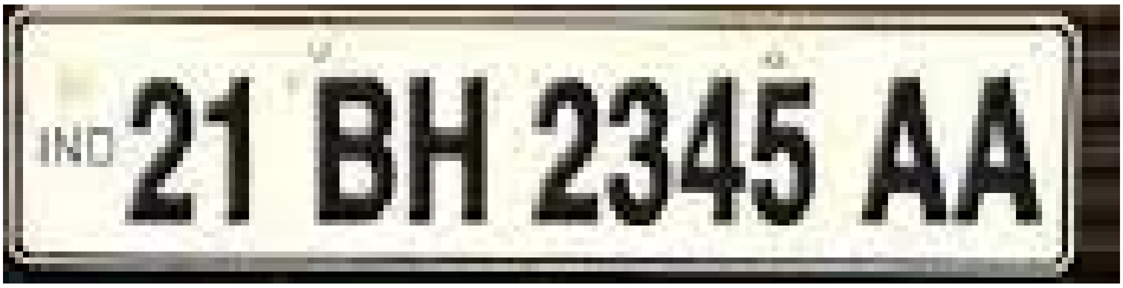
In [ ]:
```
display(output_img2, 'detected license plate in the input image')
```

detected license plate in the input image



In [ ]:
```
display(plate2, 'extracted license plate from the image')
```

extracted license plate from the image



CONTOURS

In [ ]:
```
def find_contours(dimensions, img) :

    #contours
    cntrs, _ = cv2.findContours(img.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE
    lower_width = dimensions[0]
    upper_width = dimensions[1]
    lower_height = dimensions[2]
    upper_height = dimensions[3]
    cntrs = sorted(cntrs, key=cv2.contourArea, reverse=True)[:15]
    ii = cv2.imread('contour.jpg')
    x_cntr_list = []
    target_contours = []
    img_res = []
    for cntr in cntrs :
        intX, intY, intWidth, intHeight = cv2.boundingRect(cntr)
        if intWidth > lower_width and intWidth < upper_width and intHeight > lower_
            x_cntr_list.append(intX)

            char_copy = np.zeros((44,24))
            char = img[intY:intY+intHeight, intX:intX+intWidth]
            char = cv2.resize(char, (20, 40))

            cv2.rectangle(ii, (intX,intY), (intWidth+intX, intY+intHeight), (50,21
            plt.imshow(ii, cmap='gray')
            char = cv2.subtract(255, char)
            char_copy[2:42, 2:22] = char
            char_copy[0:2, :] = 0
            char_copy[:, 0:2] = 0
            char_copy[42:44, :] = 0
            char_copy[:, 22:24] = 0

            img_res.append(char_copy)

    plt.show()
    indices = sorted(range(len(x_cntr_list)), key=lambda k: x_cntr_list[k])
    img_res_copy = []
    for idx in indices:
        img_res_copy.append(img_res[idx])
    img_res = np.array(img_res_copy)

    return img_res
```

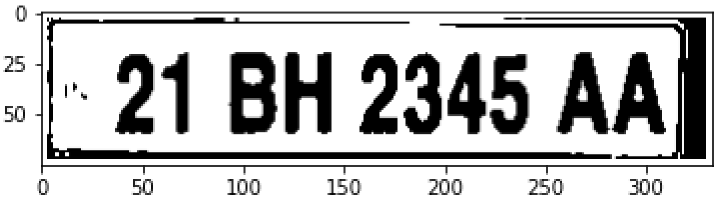
SEGMENTATION

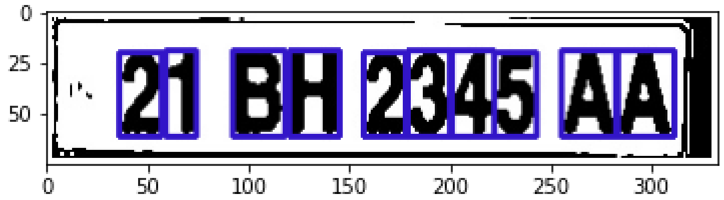
In [ ]:

```
def segment_characters(image) :

    # Preprocess cropped license plate image
    img_lp = cv2.resize(image, (333, 75))
    img_gray_lp = cv2.cvtColor(img_lp, cv2.COLOR_BGR2GRAY)
    _, img_binary_lp = cv2.threshold(img_gray_lp, 200, 255, cv2.THRESH_BINARY+cv2.
    img_binary_lp = cv2.erode(img_binary_lp, (3,3))
    img_binary_lp = cv2.dilate(img_binary_lp, (3,3))

    LP_WIDTH = img_binary_lp.shape[0]
    LP_HEIGHT = img_binary_lp.shape[1]

    # Make borders white
    img_binary_lp[0:3,:] = 255
    img_binary_lp[:,0:3] = 255
    img_binary_lp[72:75,:] = 255
    img_binary_lp[:,330:333] = 255

    # Estimations of character contours sizes of cropped license plates
    dimensions = [LP_WIDTH/6,
                       LP_WIDTH/2,
                       LP_HEIGHT/10,
                       2*LP_HEIGHT/3]
    plt.imshow(img_binary_lp, cmap='gray')
    plt.show()
    cv2.imwrite('contour.jpg',img_binary_lp)

    # Get contours within cropped license plate
    char_list = find_contours(dimensions, img_binary_lp)

    return char_list
```

In [ ]:

```
char = segment_characters(plate)
```





In [ ]:

```python
for i in range(10):
    plt.subplot(1, 10, i+1)
    plt.imshow(char[i], cmap='gray')
    plt.axis('off')
```

2
1
B
H
2
3
4
5
A
A

PREDICTION

In [ ]:

```python
def fix_dimension(img):
  new_img = np.zeros((28,28,3))
  for i in range(3):
    new_img[:,:,i] = img
  return new_img

def show_results():
    dic = {}
    characters = '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    for i,c in enumerate(characters):
        dic[i] = c

    output = []
    for i,ch in enumerate(char):
        img_ = cv2.resize(ch, (28,28), interpolation=cv2.INTER_AREA)
        img = fix_dimension(img_)
        img = img.reshape(1,28,28,3)
        y_ = model.predict_classes(img)[0] #predicting the class
        #y_ = ((model.predict(img) > 0.5).astype("int32"))[0]
        #predict_y=model.predict(img)[0]
        #y_=np.argmax(predict_y,axis=1)
        character = dic[y_]
        output.append(character)

    plate_number = ''.join(output)

    return plate_number

print(show_results())
```