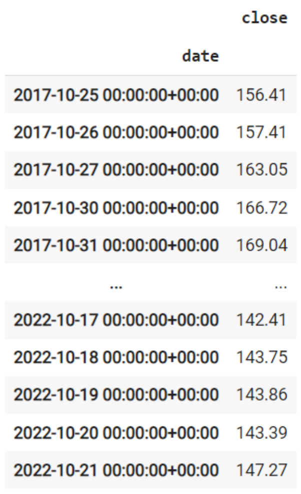
TASK - 1

Consider a large dataset (say, a time series) A. Also, consider a smaller dataset B. How do you ensure that sets A and B identify the same variable? Illustrate it with a Python script.

The objective of both the dataset is to recommend the right decision on whether to invest in a particular stock or not

Dataset A (large) is a time series data that contains the closing stock price of APPLE from 2017 to till date

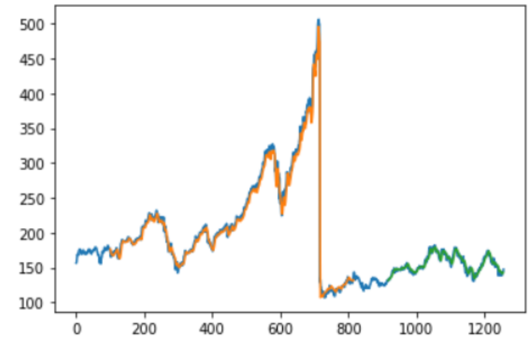


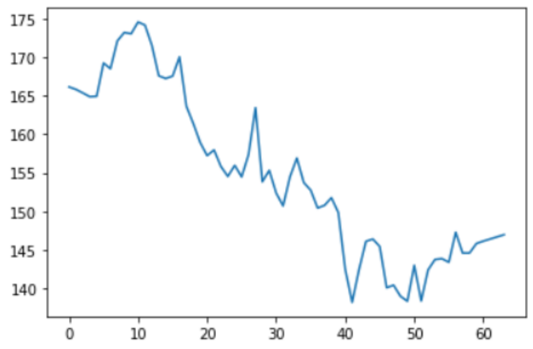
	close
date	
2017-10-25 00:00:00+00:00	156.41
2017-10-26 00:00:00+00:00	157.41
2017-10-27 00:00:00+00:00	163.05
2017-10-30 00:00:00+00:00	166.72
2017-10-31 00:00:00+00:00	169.04
...	...
2022-10-17 00:00:00+00:00	142.41
2022-10-18 00:00:00+00:00	143.75
2022-10-19 00:00:00+00:00	143.86
2022-10-20 00:00:00+00:00	143.39
2022-10-21 00:00:00+00:00	147.27

Dataset B (small) is Textual data that contains the news headlines of APPLE for the past 1 week

	ticker	date	time	title
0	AAPL	Oct-24-22	10:00AM	Stocks opened mixed ahead of Big Tech earnings
1	AAPL	Oct-24-22	09:25AM	Tech giants set to report earnings this week: ...
2	AAPL	Oct-24-22	09:05AM	Alphabet, Microsoft, Meta Platforms, Apple and...
3	AAPL	Oct-24-22	09:00AM	Wolfspeed (WOLF) to Report Q1 Earnings: What's...
4	AAPL	Oct-24-22	09:00AM	Apple 10th Gen iPad Review: Modern Build, Mode...
...	...	...	...	...
95	AAPL	Oct-18-22	11:00AM	Apple Introduces the Powerful Next-Generation ...
96	AAPL	Oct-18-22	11:00AM	Apple Introduces Next-Generation iPad Pro, Sup...
97	AAPL	Oct-18-22	10:43AM	Apple workers strike, Boeing to review 737 Max...
98	AAPL	Oct-18-22	09:44AM	12 Best Fortune 500 Stocks to Buy Now
99	AAPL	Oct-18-22	08:26AM	Foxconn: electric car supply chain highlight r...

Using the LSTM model stock prices for the next 7 days are forecasted



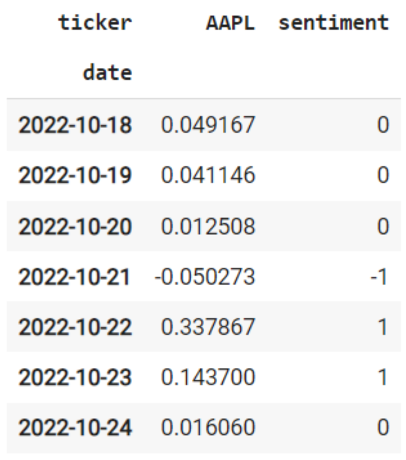


```
forecast
```

```
array([[144.5887062 ],
       [144.5887062 ],
       [145.82093178],
       [146.13185053],
       [146.40979832],
       [146.68497374],
       [146.96254077]])
```

If the change in the price of the next 7 days' closing price compared to the current closing price is positive then the model suggests that it is safe to invest or else there may be a risk involved in investing.

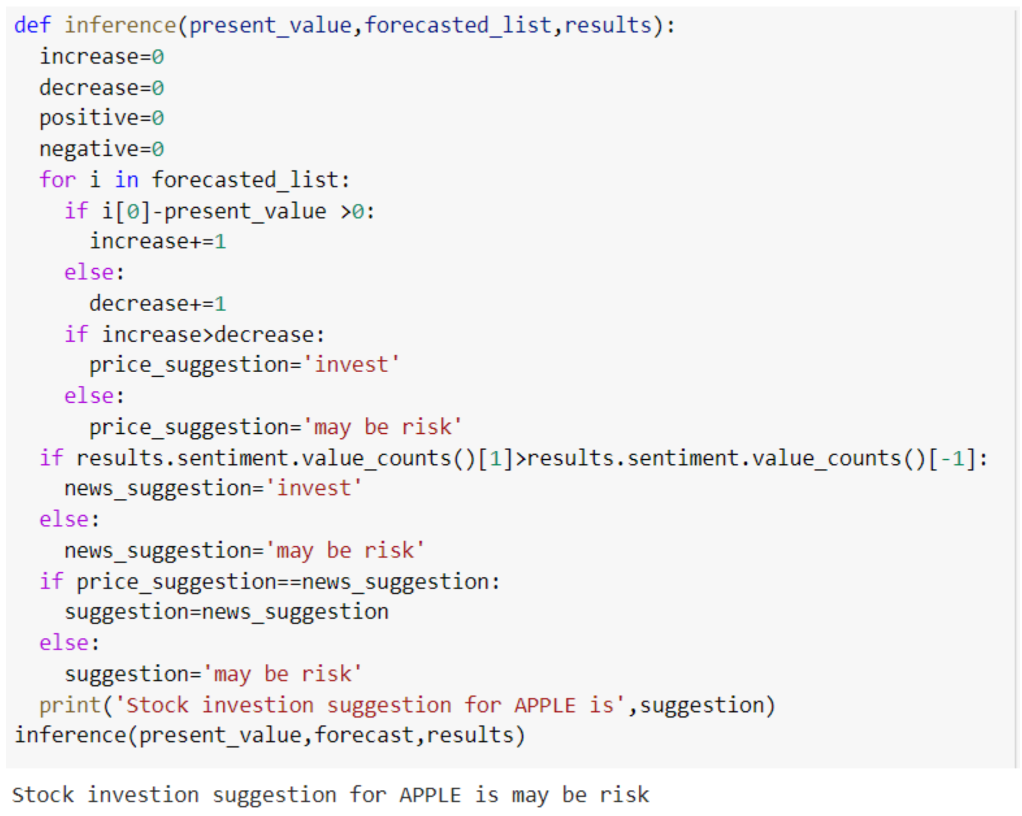
Using the sentiment intensity analyzer the sentiment of the past 7 days is calculated



ticker	AAPL	sentiment
date		
2022-10-18	0.049167	0
2022-10-19	0.041146	0
2022-10-20	0.012508	0
2022-10-21	-0.050273	-1
2022-10-22	0.337867	1
2022-10-23	0.143700	1
2022-10-24	0.016060	0

If the number of positive sentiments is greater than the number of negative sentiments then the model suggests investing in the stock or else there may be a risk involved in investing.

Inference



```python
def inference(present_value,forecasted_list,results):
  increase=0
  decrease=0
  positive=0
  negative=0
  for i in forecasted_list:
    if i[0]-present_value >0:
      increase+=1
    else:
      decrease+=1
    if increase>decrease:
      price_suggestion='invest'
    else:
      price_suggestion='may be risk'
  if results.sentiment.value_counts()[1]>results.sentiment.value_counts()[-1]:
    news_suggestion='invest'
  else:
    news_suggestion='may be risk'
  if price_suggestion==news_suggestion:
    suggestion=news_suggestion
  else:
    suggestion='may be risk'
  print('Stock investion suggestion for APPLE is',suggestion)
inference(present_value,forecast,results)
```

```
Stock investion suggestion for APPLE is may be risk
```

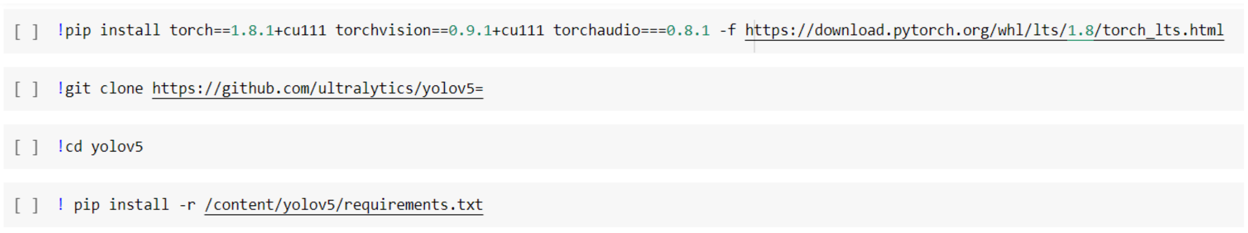
By comparing the two different datasets to identify the same target variable it is possible to incur that
If both model results are positive then the model suggests investing in the stock.

If both models give opposite results then the model suggests there is risk involved in investing in the stock.

TASK - 2

Collect data (images) and annotate them for two classes: Person and vehicle. You may use platforms such as LabelImg for annotations. You may limit it to 800 images for the dataset. Perform object detection on your collected dataset and find the mean distance between the two classes in each image. You may use YOLOv5 for detection.

Yolo Setup



```
[ ] !pip install torch==1.8.1+cu111 torchvision==0.9.1+cu111 torchaudio===0.8.1 -f https://download.pytorch.org/whl/lts/1.8/torch_lts.html
```

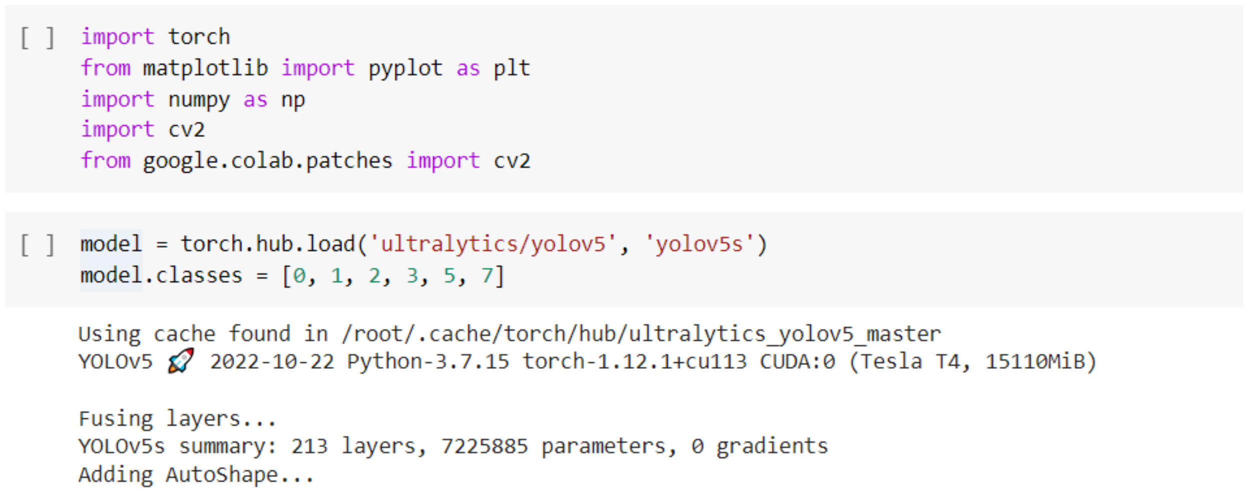
```
[ ] !git clone https://github.com/ultralytics/yolov5=
```

```
[ ] !cd yolov5
```

```
[ ] ! pip install -r /content/yolov5/requirements.txt
```

Importing the necessary libraries and modeling yolov5s

Importing the necessary libraries
Using the yolov5s version
As we want only to detect persons and vehicles:
Limiting the classes to 0 (person), 1(bicycle), 2(car), 3(motorcycles), 5(bus), 7(truck) based on the COCO dataset.



```
[ ] import torch
    from matplotlib import pyplot as plt
    import numpy as np
    import cv2
    from google.colab.patches import cv2
```

```
[ ] model = torch.hub.load('ultralytics/yolov5', 'yolov5s')
    model.classes = [0, 1, 2, 3, 5, 7]
```

```
Using cache found in /root/.cache/torch/hub/ultralytics_yolov5_master
YOLOv5 🚀 2022-10-22 Python-3.7.15 torch-1.12.1+cu113 CUDA:0 (Tesla T4, 15110MiB)

Fusing layers...
YOLOv5s summary: 213 layers, 7225885 parameters, 0 gradients
Adding AutoShape...
```

Testing the model on an image

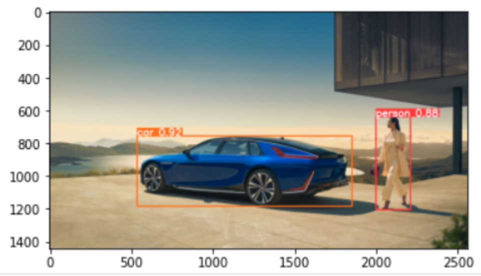
```
img ='https://media.architecturaldigest.com/photos/634d7f55f51820e32fae5c81/16:9/w_2560%2Cc_limit/49_CELESTIQ_Rear3q.jpg'
```

```
results = model(img)
results.print()
```

```
image 1/1: 1440x2560 1 person, 1 car
Speed: 338.4ms pre-process, 3481.3ms inference, 45.2ms NMS per image at shape (1, 3, 384, 640)
```

```
#grouping vehicle class
# res = results.pandas().xyxy[0]
# rn ={'car' : 'vehicle', 'bike' : 'vehicle', 'bus' : 'vehicle', 'bicycle' : 'vehicle', 'truck' : 'vehicle', 'person' : 'person'}
# res['name'] = res['name'].map(rn)
```

```
%matplotlib inline
plt.imshow(np.squeeze(results.render()))
plt.show()
```



The model is able to localize vehicles and people perfectly

Calculating mean distances between the classes

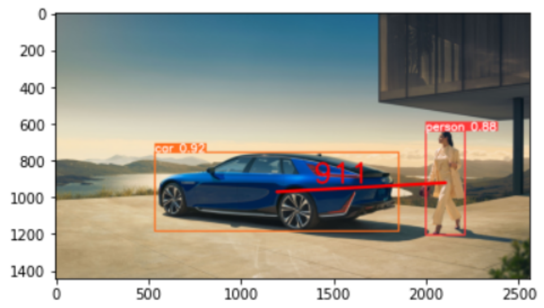
The midpoint of the objects is calculated and the mean distances of these two points are calculated

```
def midpoint(xmin, ymin, xmax, ymax):
  center_w = xmax - xmin
  center_h = ymax - ymin
  center_x = 0.5*(xmin + xmax)
  center_y = 0.5*(ymin + ymax)
  return (center_x, center_y)
center = []
for i in range(len(res)):
  center.append(midpoint(res['xmin'][i], res['ymin'][i], res['xmax'][i], res['ymax'][i]))
distance = pow(pow((center[0][0]-center[1][0]),2) + pow((center[0][1]-center[1][1]),2),0.5)
```

Plotting the results

The distance between the two objects is plotted on the image

```
image = np.squeeze(results.render())
window_name = 'Image'
start_point = (int(center[0][0]), int(center[0][1]))

end_point = (int(center[1][0]), int(center[1][1]))

# Blue color in BGR
color = (255, 0, 0)
# Line thickness
thickness = 15
label = str(int(distance))
text_org = (int(center[0][0] + 200) , int(center[0][1]) - 50)
image = cv2.line(image, start_point, end_point, color, thickness)
image = cv2.putText(image, label, text_org, cv2.FONT_HERSHEY_SIMPLEX, 5, color, 12,  cv2.LINE_AA)
# Displaying the image
plt.imshow(image)
plt.show()
```

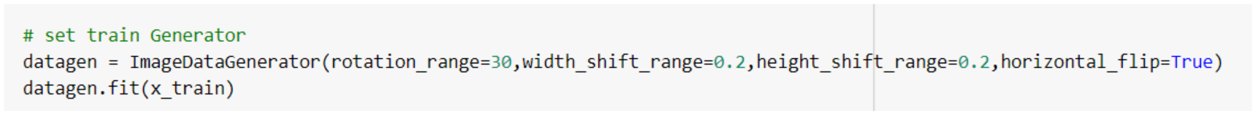


TASK - 3

Download an image dataset of your choice for binary class classification. Perform the data augmentation techniques like flipping, rotation, and transformation. Apply at least two object classification techniques both on the augmented as well as on the original dataset. Display the performance of the Algorithms. Prepare a comparison chart.

For the image dataset - The Car vs Truck classification is chosen

Data Augmentation

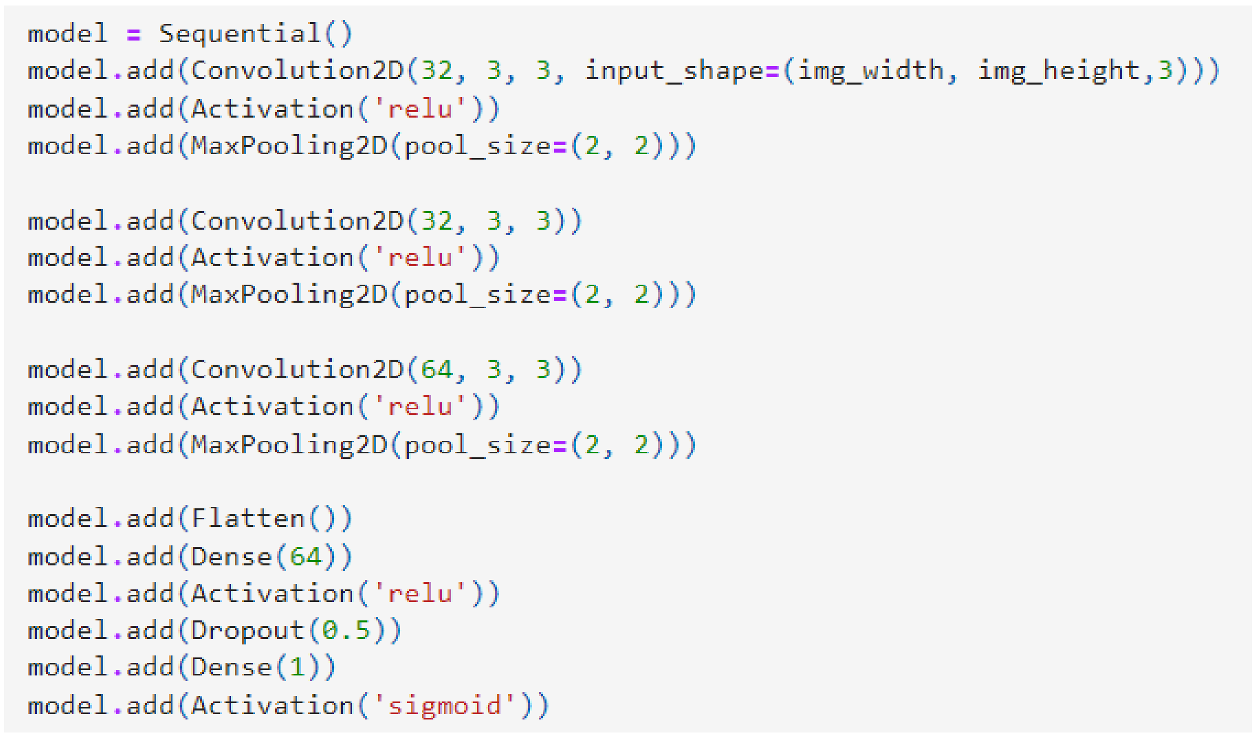


```
# set train Generator
datagen = ImageDataGenerator(rotation_range=30,width_shift_range=0.2,height_shift_range=0.2,horizontal_flip=True)
datagen.fit(x_train)
```

For Image Classification, two models were considered

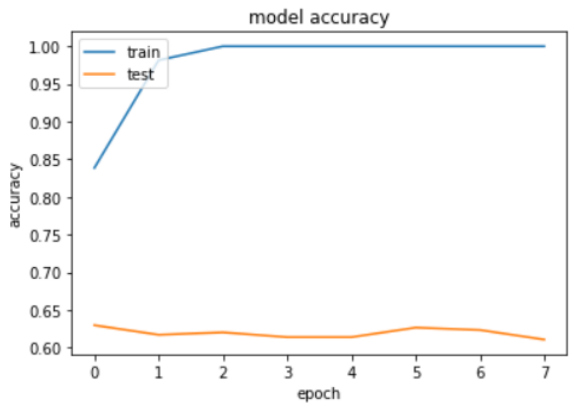
- Baseline convolutional neural network
- Transfer learning - ResNet50
-

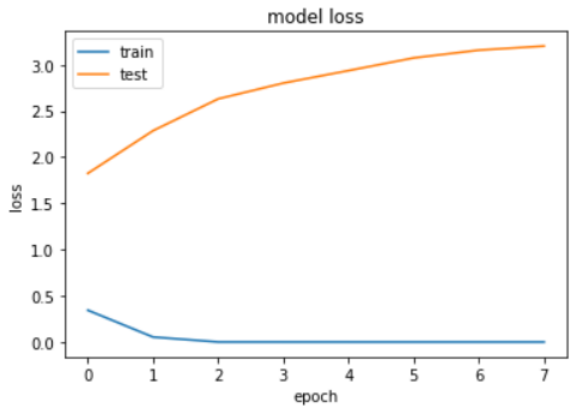
Baseline convolutional neural network



```
model = Sequential()
model.add(Convolution2D(32, 3, 3, input_shape=(img_width, img_height,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Convolution2D(32, 3, 3))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Convolution2D(64, 3, 3))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation('sigmoid'))
```

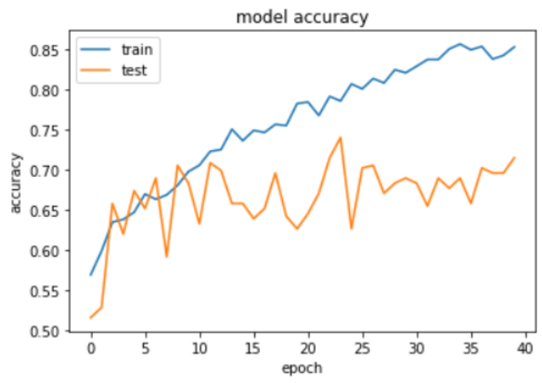
Without Augmentation

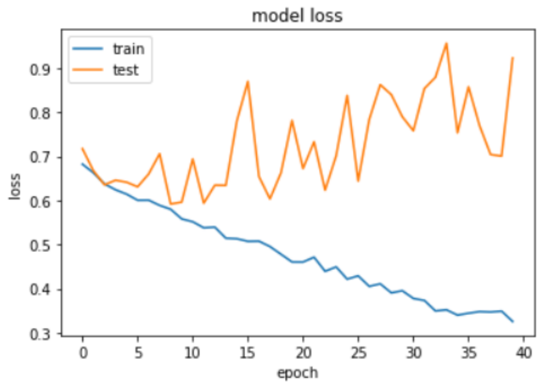




From the graph it is very clear that the model fits well for train data and not very well for test data, this is a case of overfitting

With Augmentation





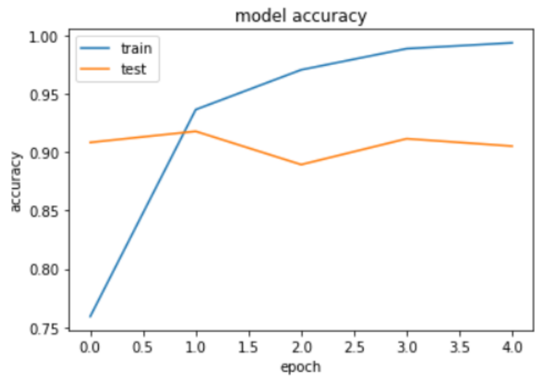
The model was able to establish a relationship without overfitting

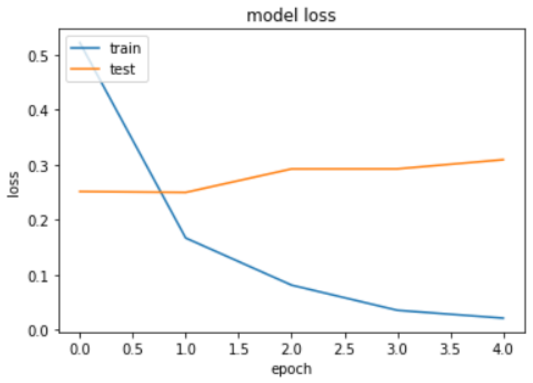
Transfer learning

ResNet50 allows the model to skip one or more layers. This approach makes it possible to train the network on many layers without affecting performance.

```
model3 = build_ResNet50(input_tensor_shape)

model3.summary()
model3.compile(optimizer=SGD(lr=0.0001, momentum=0.9), loss='binary_crossentropy', metrics=['accuracy'])
```

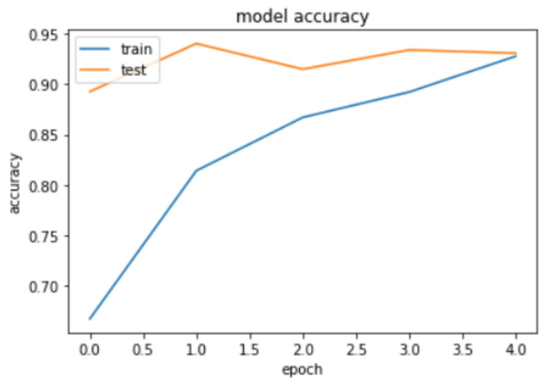
Without Augmentation

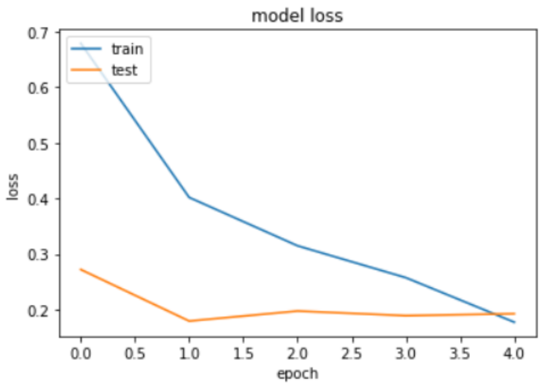




The model overfits the data in very few epochs

With Augmentation





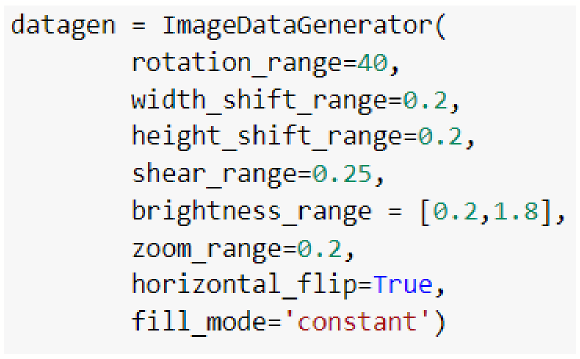
The model is able to fit the data very well

TASK - 4

Collect images of vehicles with license plates written in Indian regional languages (eg. Hindi, Kannada, Tamil, Telugu, Bengali, etc.). Apply Image augmentation techniques to the collected images. Maintain separate folders for different language license plates. You may limit to 800 images in the dataset including the augmented images.

The images were randomly collected from different sites,
Hindi - 10
Kannada - 11
Tamil - 9
Bengali - 600 (considered 10)

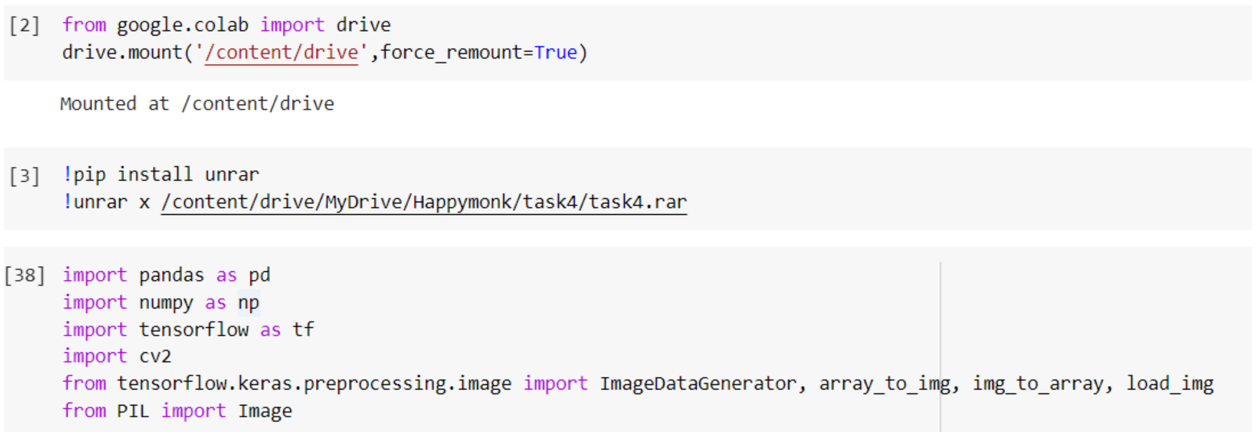
These images were augmented based on



```
datagen = ImageDataGenerator(
        rotation_range=40,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.25,
        brightness_range = [0.2,1.8],
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='constant')
```

Augmenting a total of 20 images on every image available in each class

Loading the dataset and importing the necessary libraries

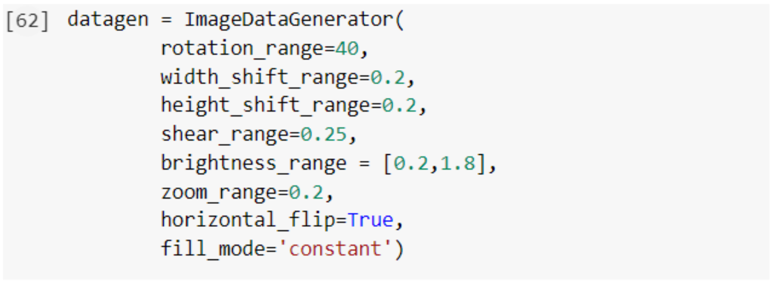


```
[2]  from google.colab import drive
     drive.mount('/content/drive',force_remount=True)

     Mounted at /content/drive
```

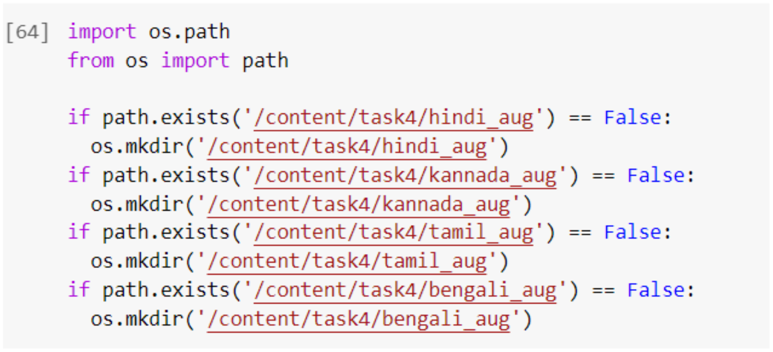
```
[3]  !pip install unrar
     !unrar x /content/drive/MyDrive/Happymonk/task4/task4.rar
```

```
[38] import pandas as pd
     import numpy as np
     import tensorflow as tf
     import cv2
     from tensorflow.keras.preprocessing.image import ImageDataGenerator, array_to_img, img_to_array, load_img
     from PIL import Image
```

Setting up the Imagedatagenerator and creating separate folders for each license plate category

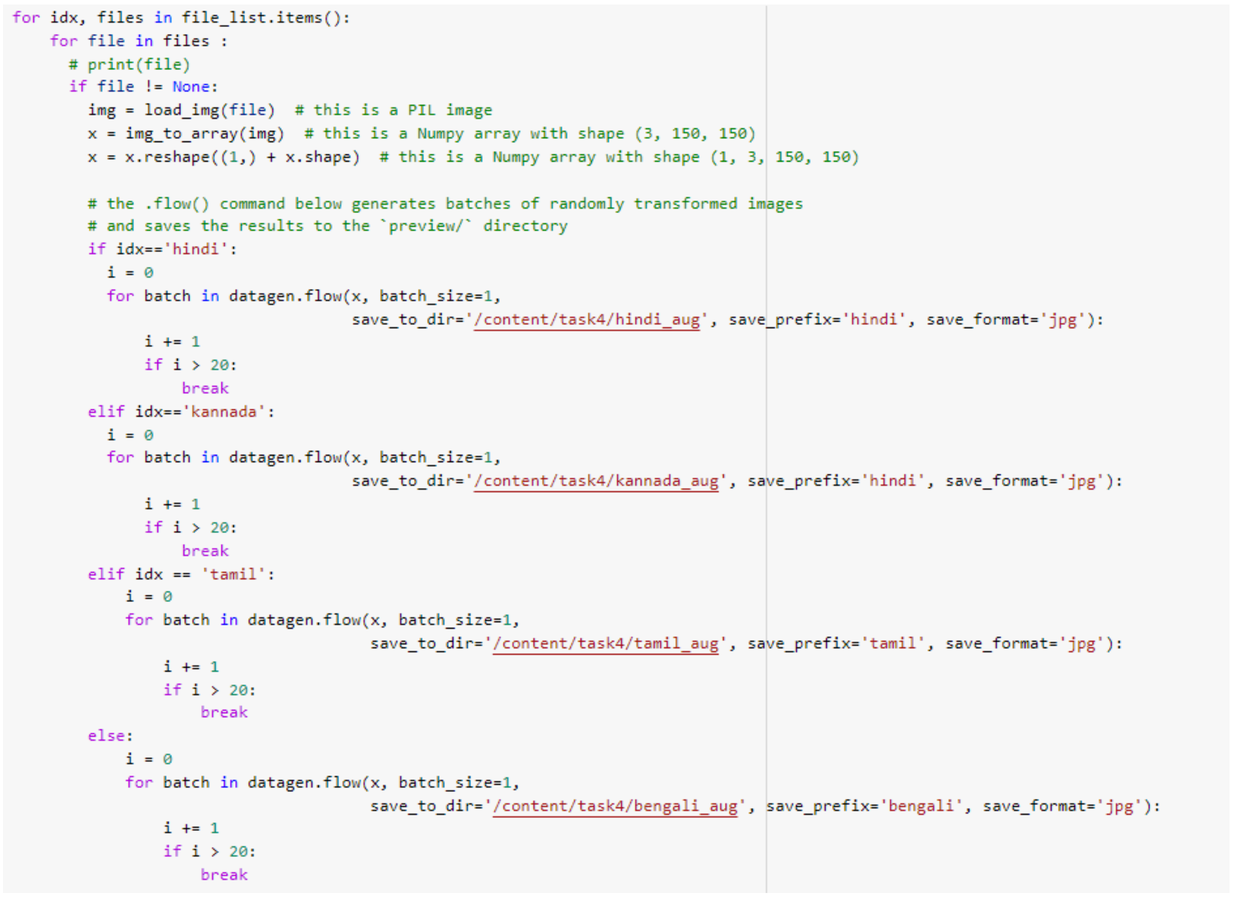


```
[62] datagen = ImageDataGenerator(
             rotation_range=40,
             width_shift_range=0.2,
             height_shift_range=0.2,
             shear_range=0.25,
             brightness_range = [0.2,1.8],
             zoom_range=0.2,
             horizontal_flip=True,
             fill_mode='constant')
```



```
[64] import os.path
     from os import path

     if path.exists('/content/task4/hindi_aug') == False:
       os.mkdir('/content/task4/hindi_aug')
     if path.exists('/content/task4/kannada_aug') == False:
       os.mkdir('/content/task4/kannada_aug')
     if path.exists('/content/task4/tamil_aug') == False:
       os.mkdir('/content/task4/tamil_aug')
     if path.exists('/content/task4/bengali_aug') == False:
       os.mkdir('/content/task4/bengali_aug')
```

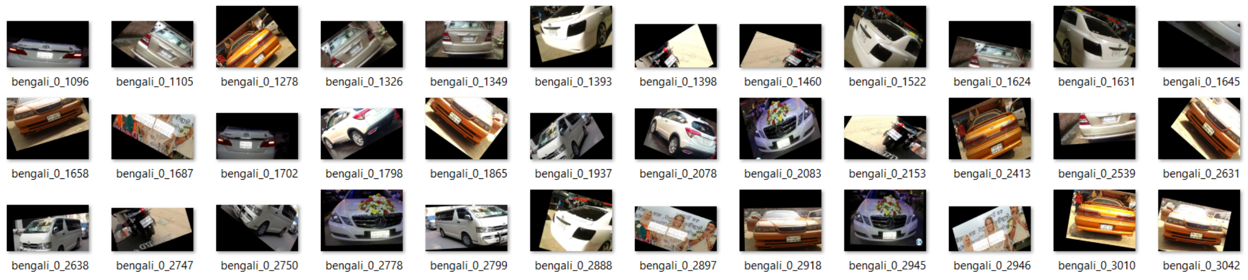
Augmenting the images



```
for idx, files in file_list.items():
    for file in files :
      # print(file)
      if file != None:
        img = load_img(file)  # this is a PIL image
        x = img_to_array(img)  # this is a Numpy array with shape (3, 150, 150)
        x = x.reshape((1,) + x.shape)  # this is a Numpy array with shape (1, 3, 150, 150)

        # the .flow() command below generates batches of randomly transformed images
        # and saves the results to the `preview/` directory
        if idx=='hindi':
          i = 0
          for batch in datagen.flow(x, batch_size=1,
                                    save_to_dir='/content/task4/hindi_aug', save_prefix='hindi', save_format='jpg'):

              i += 1
              if i > 20:
                  break
        elif idx=='kannada':
          i = 0
          for batch in datagen.flow(x, batch_size=1,
                                    save_to_dir='/content/task4/kannada_aug', save_prefix='hindi', save_format='jpg'):

              i += 1
              if i > 20:
                  break
        elif idx == 'tamil':
            i = 0
            for batch in datagen.flow(x, batch_size=1,
                                      save_to_dir='/content/task4/tamil_aug', save_prefix='tamil', save_format='jpg'):

                i += 1
                if i > 20:
                    break
        else:
            i = 0
            for batch in datagen.flow(x, batch_size=1,
                                      save_to_dir='/content/task4/bengali_aug', save_prefix='bengali', save_format='jpg'):
                i += 1
                if i > 20:
                    break
```

Augmented images for each class:

Bengali



Hindi

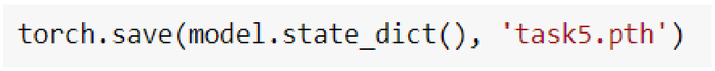
Kannada

Tamil

TASK - 5

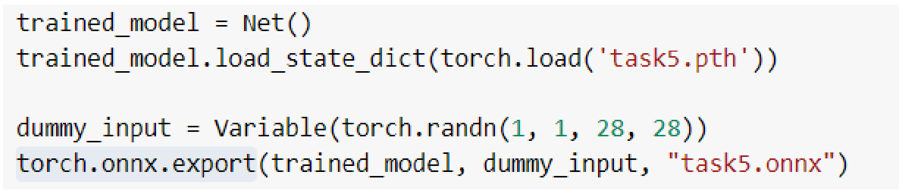
Conversion of PyTorch checkpoint file to .ONNX file

Create a simple CNN module and train it on the MNIST dataset
For simplicity 2 epochs are run
Save the model as .pth file



```
torch.save(model.state_dict(), 'task5.pth')
```

To export a model torch.onnx.export function is used - This will execute the model, recording a trace of what operators are used to compute the outputs.

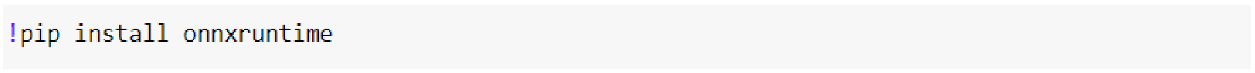


```
trained_model = Net()
trained_model.load_state_dict(torch.load('task5.pth'))

dummy_input = Variable(torch.randn(1, 1, 28, 28))
torch.onnx.export(trained_model, dummy_input, "task5.onnx")
```

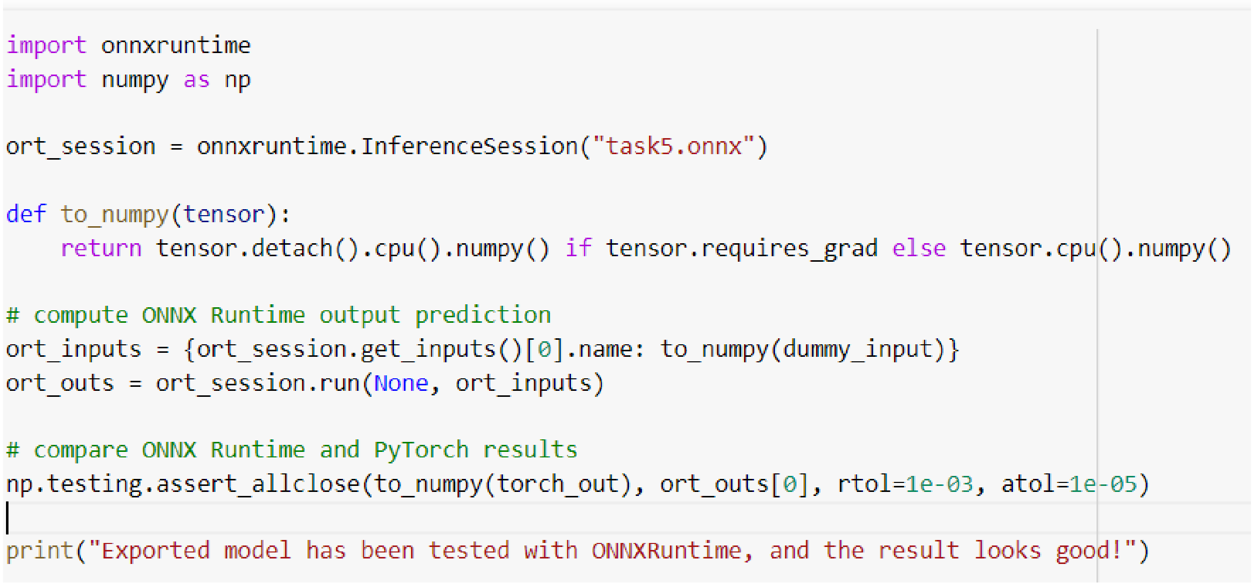
As export runs the model we provide an input tensor dummy_input

Finally, the model is exported as .ONNX file

To run the model with ONNX Runtime, we need to create an inference session
Inference on runtime



```
!pip install onnxruntime
```



```
import onnxruntime
import numpy as np

ort_session = onnxruntime.InferenceSession("task5.onnx")

def to_numpy(tensor):
    return tensor.detach().cpu().numpy() if tensor.requires_grad else tensor.cpu().numpy()

# compute ONNX Runtime output prediction
ort_inputs = {ort_session.get_inputs()[0].name: to_numpy(dummy_input)}
ort_outs = ort_session.run(None, ort_inputs)

# compare ONNX Runtime and PyTorch results
np.testing.assert_allclose(to_numpy(torch_out), ort_outs[0], rtol=1e-03, atol=1e-05)

print("Exported model has been tested with ONNXRuntime, and the result looks good!")
```