# Diabetic Retinopathy Report

## 1. Overview

Diabetic Retinopathy (DR) is an eye disease associated with long-standing diabetes. Around 40% to 45% of Americans with diabetes have some stage of the disease. A clinician has rated the presence of diabetic retinopathy in each image on a scale of 0 to 4, according to the following scale:

0 - No DR

1 - Mild

2 - Moderate

3 - Severe

4 - Proliferative DR

Our task is to create an automated analysis system capable of assigning a score based on this scale.

## 2. Dataset

Training data consists of 35126 images which are further classified into following five categories:

| **Category** | **Scaling** | **Number of images** |
|---|---|---|
| No DR | 0 | 25810 |
| Mild DR | 1 | 2443 |
| Moderate DR | 2 | 5292 |
| Severe DR | 3 | 873 |
| Proliferative DR | 4 | 708 |

## 3. Softwares

❿ Google Cloud instance  with following specifications is used:

a) Zone : us-east1-d  
b) Machine type : 24 vCPU  
c) Boot disk : Ubuntu 17.04 with 400GB size

❿ Python version : 2.7

❿ Libraries used are given in the code

# 4. Image Preprocessing and Data augmentation:

We used opencv to perform foolowing modifications in the existing data

❿ Resized the images to (385*385) pixels.
❿ Cropped horizontally the images by 40 pixels.
❿ Randomly rotated the images between 0 and 360 degrees.
❿ Subtracted the local average color, the avgerage gets mapped to 50% gray.
❿ Applied horizontal and vertical flip.

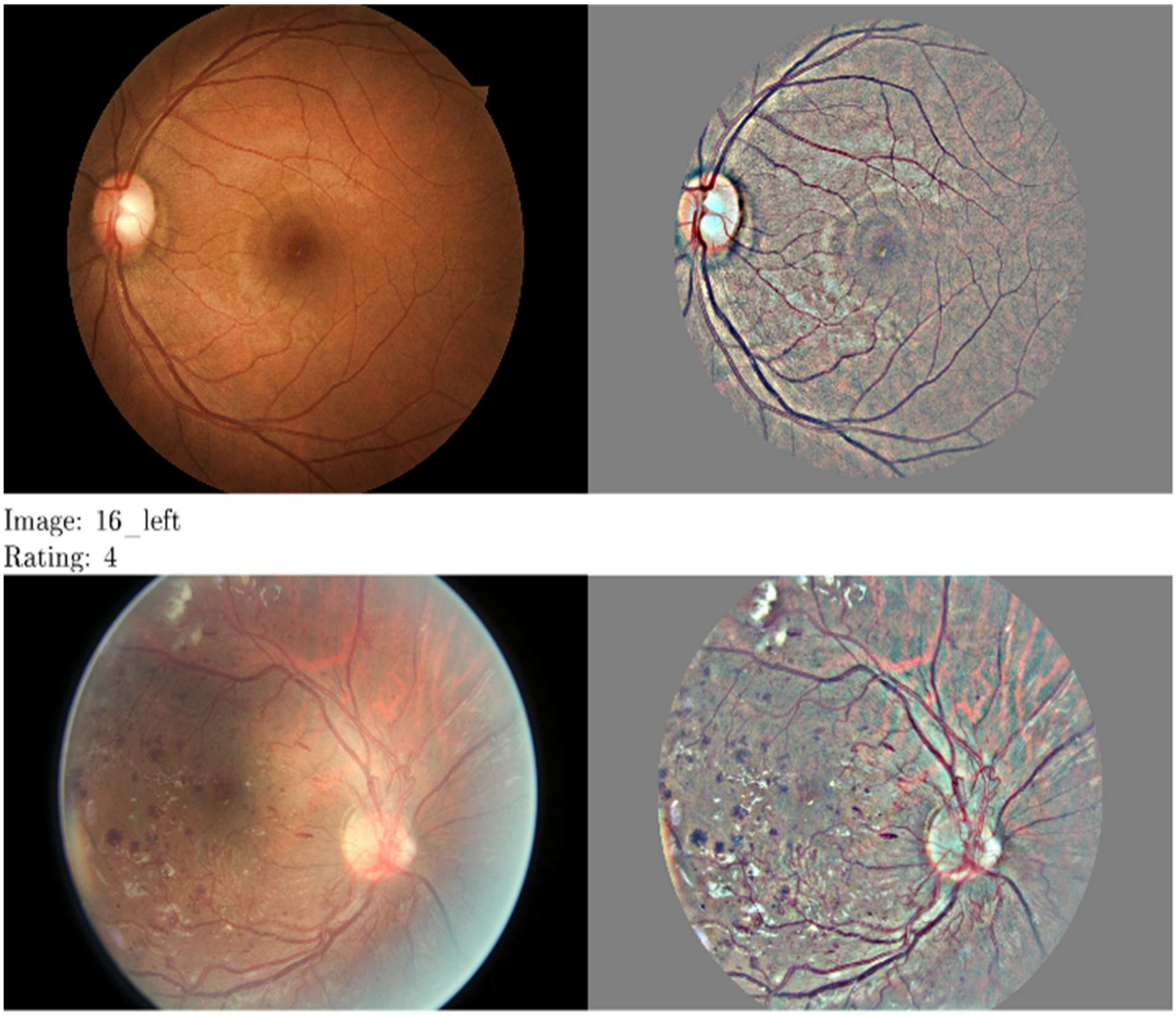
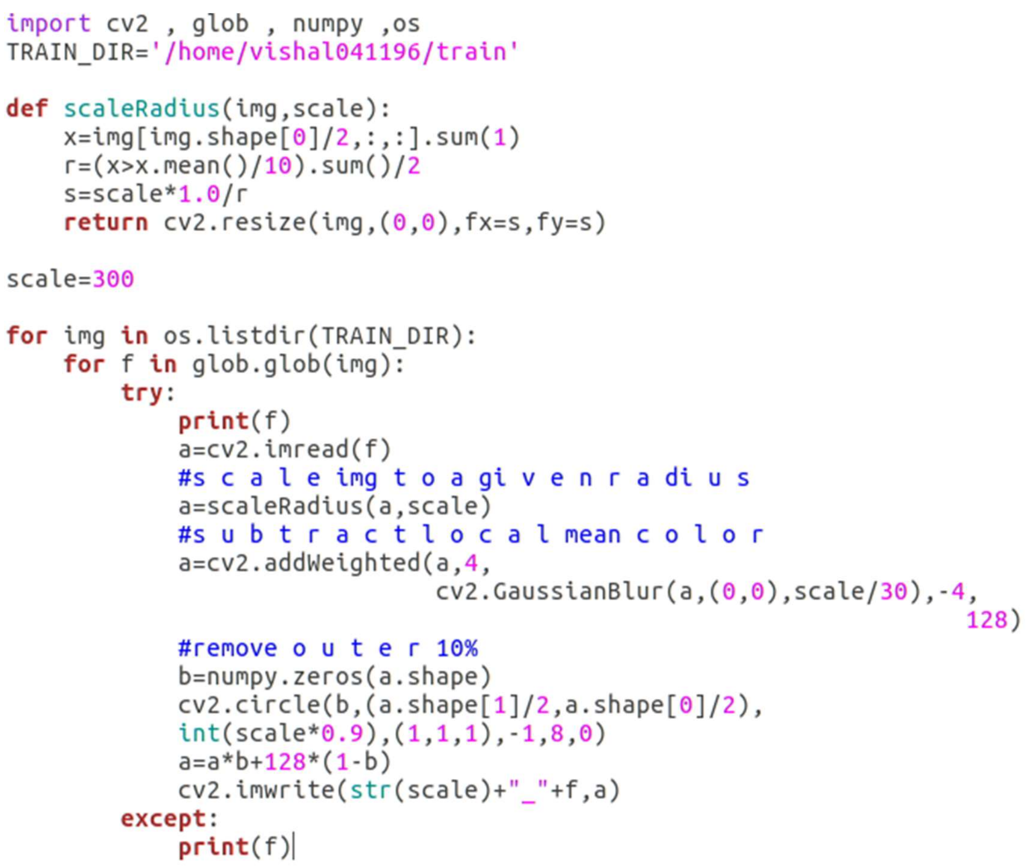


Image: 16_left
Rating: 4

Figure 4: Two images from the training set. Original images on the left and preprocessed images on the right.

❾ Code for Image Preprocessing:



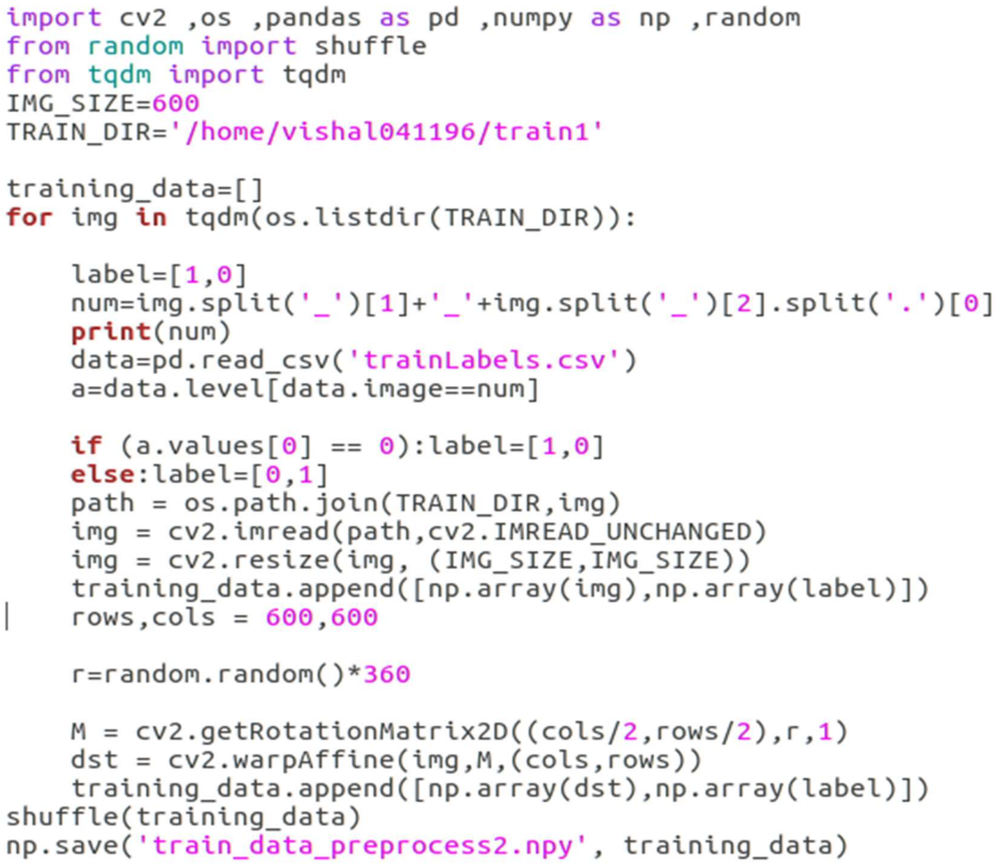
```
import cv2 , glob , numpy ,os
TRAIN_DIR='/home/vishal041196/train'

def scaleRadius(img,scale):
    x=img[img.shape[0]/2,:,:].sum(1)
    r=(x>x.mean()/10).sum()/2
    s=scale*1.0/r
    return cv2.resize(img,(0,0),fx=s,fy=s)

scale=300

for img in os.listdir(TRAIN_DIR):
    for f in glob.glob(img):
        try:
            print(f)
            a=cv2.imread(f)
            #scale img to a given radius
            a=scaleRadius(a,scale)
            #subtract local mean color
            a=cv2.addWeighted(a,4,
                              cv2.GaussianBlur(a,(0,0),scale/30),-4,
                                                                    128)
            #remove outer 10%
            b=numpy.zeros(a.shape)
            cv2.circle(b,(a.shape[1]/2,a.shape[0]/2),
            int(scale*0.9),(1,1,1),-1,8,0)
            a=a*b+128*(1-b)
            cv2.imwrite(str(scale)+"_"+f,a)
        except:
            print(f)
```

❾ Code for creating training data:



```
import cv2 ,os ,pandas as pd ,numpy as np ,random
from random import shuffle
from tqdm import tqdm
IMG_SIZE=600
TRAIN_DIR='/home/vishal041196/train1'

training_data=[]
for img in tqdm(os.listdir(TRAIN_DIR)):

    label=[1,0]
    num=img.split('_')[1]+'_'+img.split('_')[2].split('.')[0]
    print(num)
    data=pd.read_csv('trainLabels.csv')
    a=data.level[data.image==num]

    if (a.values[0] == 0):label=[1,0]
    else:label=[0,1]
    path = os.path.join(TRAIN_DIR,img)
    img = cv2.imread(path,cv2.IMREAD_UNCHANGED)
    img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
    training_data.append([np.array(img),np.array(label)])
    rows,cols = 600,600

    r=random.random()*360

    M = cv2.getRotationMatrix2D((cols/2,rows/2),r,1)
    dst = cv2.warpAffine(img,M,(cols,rows))
    training_data.append([np.array(dst),np.array(label)])
shuffle(training_data)
np.save('train_data_preprocess2.npy', training_data)
```

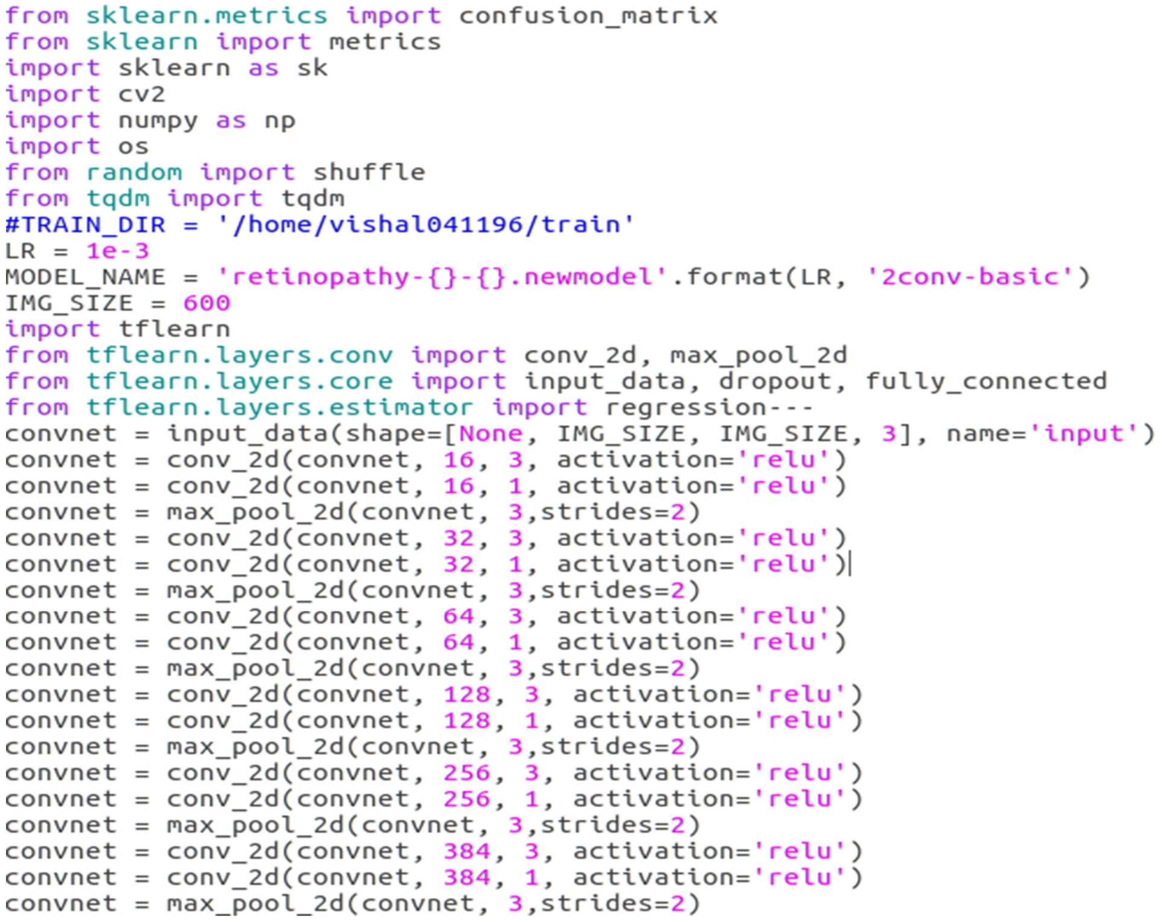
Convolution neural network model:

**Structure:-**

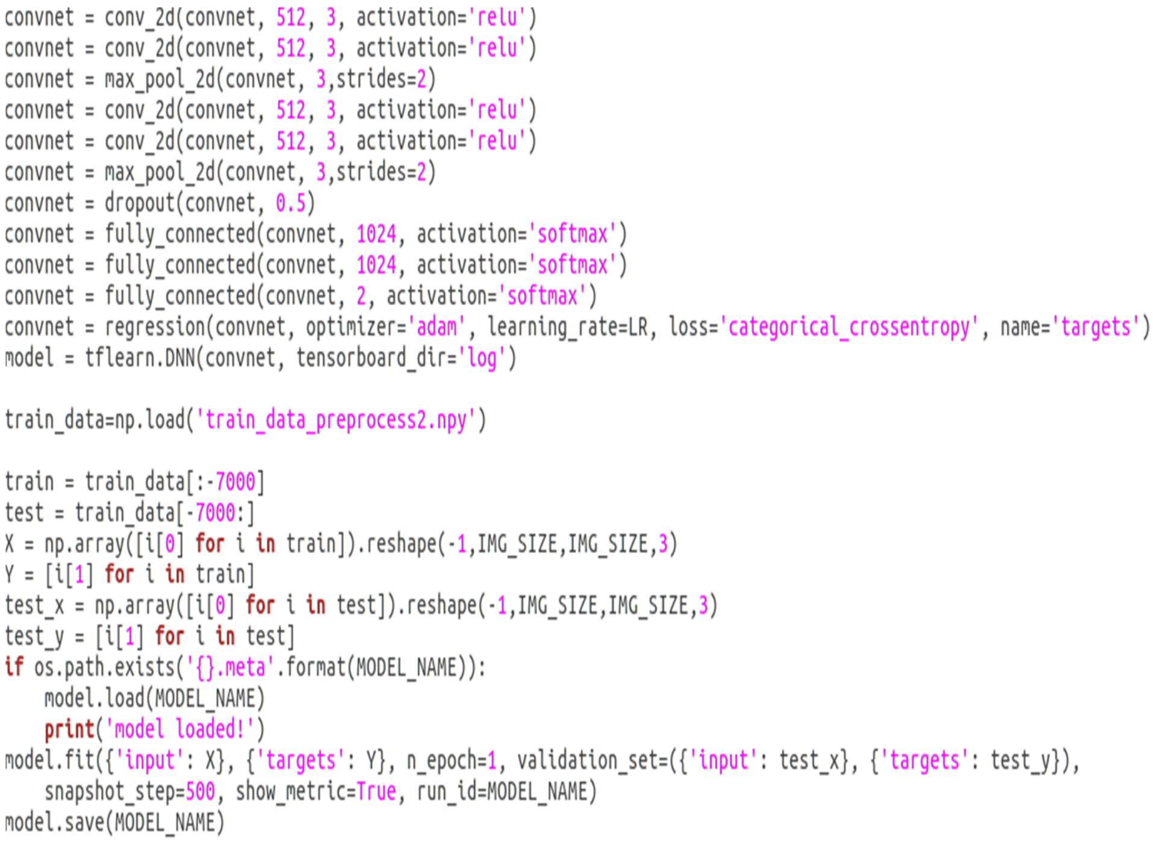
The core of our solution was a deep convolutional neural network. Although we started with fairly shallow models — 4 convolutional layers, we quickly discovered that adding more layers, and filters inside layers helps a lot. Our best single model consisted of 16 convolutional layers.
The detailed architecture is:

| **Type** | **Units** | **Filter** | **Stride** |
|---|---|---|---|
| Input | | | |
| Conv | 16 | 3 | 1 |
| Conv | 16 | 1 | 1 |
| MaxPool | - | 3 | 2 |
| Conv | 32 | 3 | 1 |
| Conv | 32 | 1 | 1 |
| MaxPool | - | 3 | 2 |
| Conv | 64 | 3 | 1 |
| Conv | 64 | 1 | 1 |
| MaxPool | - | 3 | 2 |
| Conv | 128 | 3 | 1 |
| Conv | 128 | 1 | 1 |
| MaxPool | - | 3 | 2 |
| Conv | 256 | 3 | 1 |
| Conv | 256 | 1 | 1 |
| MaxPool | - | 3 | 2 |
| Conv | 384 | 3 | 1 |
| Conv | 384 | 3 | 1 |
| MaxPool | - | 3 | 2 |
| Conv | 512 | 3 | 1 |
| Conv | 512 | 3 | 1 |
| MaxPool | - | 3 | 2 |
| Conv | 512 | 3 | 1 |
| Conv | 512 | 3 | 1 |
| MaxPool | - | 3 | 2 |
| Dropout | | 0.5 | - |
| fully_connected | 1024 | - | - |
| fully_connected | 1024 | - | - |
| fully_connected | 2 | - | - |

❾ Code for CNN Model:



```
from sklearn.metrics import confusion_matrix
from sklearn import metrics
import sklearn as sk
import cv2
import numpy as np
import os
from random import shuffle
from tqdm import tqdm
#TRAIN_DIR = '/home/vishal041196/train'
LR = 1e-3
MODEL_NAME = 'retinopathy-{}-{}.newmodel'.format(LR, '2conv-basic')
IMG_SIZE = 600
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression---
convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 3], name='input')
convnet = conv_2d(convnet, 16, 3, activation='relu')
convnet = conv_2d(convnet, 16, 1, activation='relu')
convnet = max_pool_2d(convnet, 3,strides=2)
convnet = conv_2d(convnet, 32, 3, activation='relu')
convnet = conv_2d(convnet, 32, 1, activation='relu')
convnet = max_pool_2d(convnet, 3,strides=2)
convnet = conv_2d(convnet, 64, 3, activation='relu')
convnet = conv_2d(convnet, 64, 1, activation='relu')
convnet = max_pool_2d(convnet, 3,strides=2)
convnet = conv_2d(convnet, 128, 3, activation='relu')
convnet = conv_2d(convnet, 128, 1, activation='relu')
convnet = max_pool_2d(convnet, 3,strides=2)
convnet = conv_2d(convnet, 256, 3, activation='relu')
convnet = conv_2d(convnet, 256, 1, activation='relu')
convnet = max_pool_2d(convnet, 3,strides=2)
convnet = conv_2d(convnet, 384, 3, activation='relu')
convnet = conv_2d(convnet, 384, 1, activation='relu')
convnet = max_pool_2d(convnet, 3,strides=2)
```

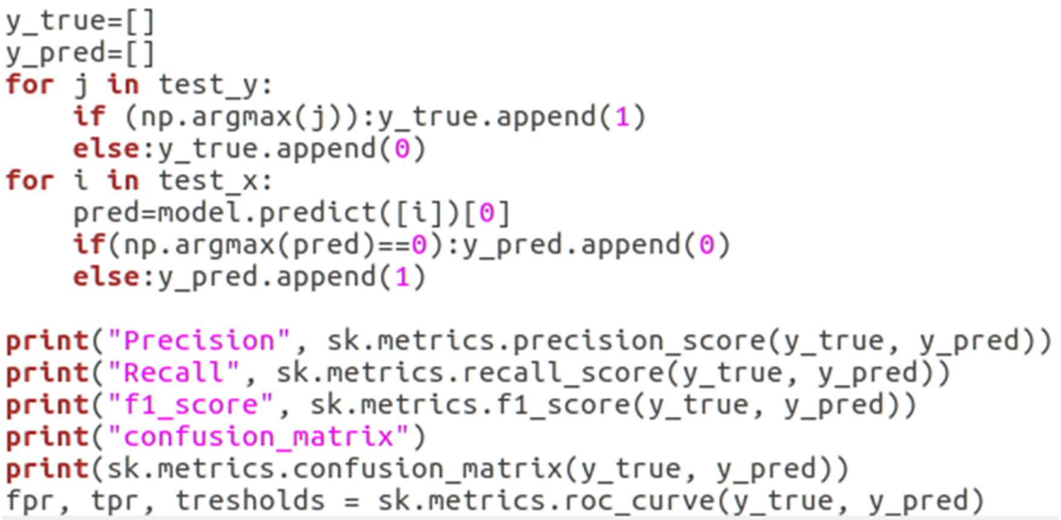


```
convnet = conv_2d(convnet, 512, 3, activation='relu')
convnet = conv_2d(convnet, 512, 3, activation='relu')
convnet = max_pool_2d(convnet, 3,strides=2)
convnet = conv_2d(convnet, 512, 3, activation='relu')
convnet = conv_2d(convnet, 512, 3, activation='relu')
convnet = max_pool_2d(convnet, 3,strides=2)
convnet = dropout(convnet, 0.5)
convnet = fully_connected(convnet, 1024, activation='softmax')
convnet = fully_connected(convnet, 1024, activation='softmax')
convnet = fully_connected(convnet, 2, activation='softmax')
convnet = regression(convnet, optimizer='adam', learning_rate=LR, loss='categorical_crossentropy', name='targets')
model = tflearn.DNN(convnet, tensorboard_dir='log')

train_data=np.load('train_data_preprocess2.npy')

train = train_data[:-7000]
test = train_data[-7000:]
X = np.array([i[0] for i in train]).reshape(-1,IMG_SIZE,IMG_SIZE,3)
Y = [i[1] for i in train]
test_x = np.array([i[0] for i in test]).reshape(-1,IMG_SIZE,IMG_SIZE,3)
test_y = [i[1] for i in test]
if os.path.exists('{}.meta'.format(MODEL_NAME)):
    model.load(MODEL_NAME)
    print('model loaded!')
model.fit({'input': X}, {'targets': Y}, n_epoch=1, validation_set=({'input': test_x}, {'targets': test_y}),
    snapshot_step=500, show_metric=True, run_id=MODEL_NAME)
model.save(MODEL_NAME)
```

❾ Code for checking the predictions:



```
y_true=[]
y_pred=[]
for j in test_y:
    if (np.argmax(j)):y_true.append(1)
    else:y_true.append(0)
for i in test_x:
    pred=model.predict([i])[0]
    if(np.argmax(pred)==0):y_pred.append(0)
    else:y_pred.append(1)

print("Precision", sk.metrics.precision_score(y_true, y_pred))
print("Recall", sk.metrics.recall_score(y_true, y_pred))
print("f1_score", sk.metrics.f1_score(y_true, y_pred))
print("confusion_matrix")
print(sk.metrics.confusion_matrix(y_true, y_pred))
fpr, tpr, tresholds = sk.metrics.roc_curve(y_true, y_pred)
```

## Results and Conclusions:

Number of images in Training set = 64890
Number of images in Validation set = 7500

After 70 epochs the results are as follows:

Training accuracy = .9615
Validation accuracy = .7299
Precision = .753
Recall = .6909
f1_score = .7297

Confusion matrix :

| | |
|---|---|
| 2860 | 857 |
| 1160 | 2614 |