## B9DA103 Data Mining

### Assignment – 2

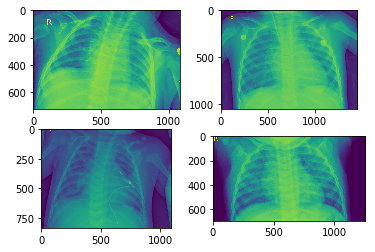
# CRISP-DM Data Preparation, Modelling, Evaluation

Prepared By: Prateek Dixit (10503786) Mohammad Asjad (10503994) \*\*\*

## Sample Data

# Imports to view data  
from glob import glob  
import matplotlib.pyplot as plt  
import matplotlib.image as mpimg  
from numpy import floor  
import random  
  
print("Samples images for PNEUMONIA X-Rays")  
path = 'D:/data/ChestXRay2017/chest\_xray/train/PNEUMONIA/\*\*'  
path\_contents = glob(path)  
plt.figure(figsize=(16,16))  
imgs = random.sample(path\_contents, 4)  
f, axarr = plt.subplots(2, 2)  
axarr[0, 0].imshow(mpimg.imread(imgs[0]))  
axarr[0, 1].imshow(mpimg.imread(imgs[1]))  
axarr[1, 0].imshow(mpimg.imread(imgs[2]))  
axarr[1, 1].imshow(mpimg.imread(imgs[3]))

Samples images for PNEUMONIA X-Rays  
  
  
  
  
  
<matplotlib.image.AxesImage at 0x1d93faadb00>  
  
  
  
  
<Figure size 1152x1152 with 0 Axes>



png

## Data Preprocessing and Setup

# Importing the Keras libraries and packages  
from keras.models import Sequential  
from keras.layers import Conv2D  
from keras.layers import MaxPooling2D  
from keras.layers import Flatten  
from keras.layers import Dense  
  
from keras.preprocessing.image import ImageDataGenerator  
  
train\_datagen = ImageDataGenerator(rescale = 1./255,  
 shear\_range = 0.2,  
 zoom\_range = 0.2,  
 horizontal\_flip = True)  
  
test\_datagen = ImageDataGenerator(rescale = 1./255)  
  
train = train\_datagen.flow\_from\_directory('D:/data/ChestXRay2017/chest\_xray/train',  
 target\_size = (64, 64),  
 batch\_size = 32,  
 class\_mode = 'binary')  
  
test = test\_datagen.flow\_from\_directory('D:/data/ChestXRay2017/chest\_xray/test',  
 target\_size = (64, 64),  
 batch\_size = 32,  
 class\_mode = 'binary')

Using TensorFlow backend.  
  
  
Found 5232 images belonging to 2 classes.  
Found 624 images belonging to 2 classes.

## Initializing the CNN

classifier = Sequential()  
  
# Step 1 - Convolution  
classifier.add(Conv2D(32, (3, 3), input\_shape = (64, 64, 3), activation = 'relu'))  
  
# Step 2 - Pooling  
classifier.add(MaxPooling2D(pool\_size = (2, 2)))  
  
# Adding a second convolutional layer  
classifier.add(Conv2D(32, (3, 3), activation = 'relu'))  
classifier.add(MaxPooling2D(pool\_size = (2, 2)))  
  
# Step 3 - Flattening  
classifier.add(Flatten())  
  
# Step 4 - Full connection  
classifier.add(Dense(units = 128, activation = 'relu'))  
classifier.add(Dense(units = 1, activation = 'sigmoid'))  
  
# Compiling the CNN  
classifier.compile(optimizer = 'adam', loss = 'binary\_crossentropy', metrics = ['accuracy'])

WARNING:tensorflow:From C:\Users\Asjad\Anaconda3\lib\site-packages\tensorflow\python\framework\op\_def\_library.py:263: colocate\_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.  
Instructions for updating:  
Colocations handled automatically by placer.

### The First and Second hidden layers had 32 filters and uses Rectifier Linear Unit activation

## Model Fitting

classifier.fit\_generator(train,  
 steps\_per\_epoch = 5216,  
 epochs = 10,  
 validation\_data = test,  
 validation\_steps = 624)

WARNING:tensorflow:From C:\Users\Asjad\Anaconda3\lib\site-packages\tensorflow\python\ops\math\_ops.py:3066: to\_int32 (from tensorflow.python.ops.math\_ops) is deprecated and will be removed in a future version.  
Instructions for updating:  
Use tf.cast instead.  
Epoch 1/10  
5216/5216 [==============================] - 5058s 970ms/step - loss: 0.1361 - acc: 0.9473 - val\_loss: 0.2959 - val\_acc: 0.9150  
Epoch 2/10  
5216/5216 [==============================] - 4735s 908ms/step - loss: 0.0702 - acc: 0.9740 - val\_loss: 0.3144 - val\_acc: 0.9041  
Epoch 3/10  
5216/5216 [==============================] - 4269s 818ms/step - loss: 0.0469 - acc: 0.9827 - val\_loss: 0.3946 - val\_acc: 0.9082  
Epoch 4/10  
5216/5216 [==============================] - 4344s 833ms/step - loss: 0.0319 - acc: 0.9883 - val\_loss: 0.6945 - val\_acc: 0.8782  
Epoch 5/10  
5216/5216 [==============================] - 5853s 1s/step - loss: 0.0228 - acc: 0.9915 - val\_loss: 0.6644 - val\_acc: 0.8910  
Epoch 6/10  
5216/5216 [==============================] - 6767s 1s/step - loss: 0.0179 - acc: 0.9938 - val\_loss: 0.5829 - val\_acc: 0.8944  
Epoch 7/10  
5216/5216 [==============================] - 6377s 1s/step - loss: 0.0148 - acc: 0.9948 - val\_loss: 0.5871 - val\_acc: 0.9018  
Epoch 8/10  
5216/5216 [==============================] - 6369s 1s/step - loss: 0.0132 - acc: 0.9953 - val\_loss: 0.6021 - val\_acc: 0.9039  
Epoch 9/10  
5216/5216 [==============================] - 4535s 869ms/step - loss: 0.0106 - acc: 0.9962 - val\_loss: 0.7347 - val\_acc: 0.8912  
Epoch 10/10  
5216/5216 [==============================] - 6207s 1s/step - loss: 0.0095 - acc: 0.9966 - val\_loss: 0.4764 - val\_acc: 0.9167  
  
  
  
  
  
<keras.callbacks.History at 0x156e46a8828>

### It took approximately 15 Hours 8 Minutes to train the model while running on 8GB RAM and Intel(R)Core(TM) i5-3360M CPU @ 2.80GHz

## Prediction

import numpy as np  
from glob import glob  
import random  
  
from keras.preprocessing import image  
  
print("Samples Prediction of PNEUMONIA X-Rays")  
path = 'D:/data/ChestXRay2017/chest\_xray/test/PNEUMONIA/\*\*'  
path\_content = glob(path)  
print(len(path\_content))  
print(path.split('/')[-2])  
predctr = 0  
  
for p in path\_content:  
# img = random.sample(p, 1)  
   
  
 test\_image = image.load\_img(p, target\_size = (64, 64))  
 test\_image = image.img\_to\_array(test\_image)  
 test\_image = np.expand\_dims(test\_image, axis = 0)  
 result = classifier.predict(test\_image)  
 if result[0][0] == 1:  
 prediction = 'PNEUMONIA'  
 predctr +=1  
 else:  
 prediction = 'NORMAL'  
  
print(f'''Accuracy = {predctr\*100/len(path\_content)}%''')

Samples Prediction of PNEUMONIA X-Rays  
390  
PNEUMONIA  
Accuracy = 98.46153846153847%

## Export Model

model\_json = classifier.to\_json()

with open("model.json", "w") as json\_file:  
 json\_file.write(model\_json)  
# serialize weights to HDF5  
classifier.save\_weights("model.h5")  
print("Saved model to disk")

Saved model to disk

## Load Model

# load json and create model  
from keras.models import model\_from\_json  
json\_file = open('model.json', 'r')  
loaded\_model\_json = json\_file.read()  
json\_file.close()

loaded\_model = model\_from\_json(loaded\_model\_json)  
# load weights into new model  
loaded\_model.load\_weights("model.h5")  
print("Loaded model from disk")  
   
# evaluate loaded model on test data  
loaded\_model.compile(optimizer = 'adam', loss = 'binary\_crossentropy', metrics = ['accuracy'])

Loaded model from disk

## Testing the Model

import numpy as np  
from glob import glob  
import random  
  
from keras.preprocessing import image  
  
print("Samples Prediction of PNEUMONIA X-Rays")  
path = 'D:/data/ChestXRay2017/chest\_xray/test/NORMAL/\*\*'  
path\_content = glob(path)  
print(len(path\_content))  
print(path.split('/')[-2])  
predctr = 0  
  
for p in path\_content:  
# img = random.sample(p, 1)  
   
  
 test\_image = image.load\_img(p, target\_size = (64, 64))  
 test\_image = image.img\_to\_array(test\_image)  
 test\_image = np.expand\_dims(test\_image, axis = 0)  
 result = loaded\_model.predict(test\_image)  
 if result[0][0] == 1:  
 prediction = 'PNEUMONIA'  
 predctr +=1  
 else:  
 prediction = 'NORMAL'  
  
print(f'''Accuracy = {predctr\*100/len(path\_content)}%''')

Samples Prediction of PNEUMONIA X-Rays  
234  
NORMAL  
Accuracy = 55.98290598290598%