MAIS 202 - PROJECT DELIVERABLE 2

1. Problem statement

I will be implementing a model that classify X-Ray scans from patients with pneumonia. I will use the dataset from Kaggle. https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia

2. Data Preprocessing

Firstly, I read the images from 3 folders (Train, Val, Test). Each of them contains chest images of people who have pneumonia and people who don't. Since the input dimensions of the images are too big, I applied image augmentation and changed the size of the images. Finally, I plotted the images corresponding to their labels of NORMAL and PNEUMONIA.

```
[7]  1 # all images in the dataset
     2 train_folder= 'chest_xray/chest_xray/train/'
     3 val_folder = 'chest_xray/chest_xray/val/'
     4 test_folder = 'chest_xray/chest_xray/test/'
```

```
[8]  1 # image augmentation
     2 gen = ImageDataGenerator()
     3 train_batches = gen.flow_from_directory(train_folder,target_size = (64, 64),color_mode="grayscale",shuffle=True,seed=1,
     4                                         batch_size=16)
     5 valid_batches = gen.flow_from_directory(val_folder, target_size = (64, 64),color_mode="grayscale", shuffle=True,seed=1,
     6                                         batch_size=16)
     7 test_batches = gen.flow_from_directory(test_folder, target_size = (64, 64), shuffle=False,
     8                                        color_mode="grayscale", batch_size=8)
```

```
Found 5216 images belonging to 2 classes.
Found 16 images belonging to 2 classes.
Found 624 images belonging to 2 classes.
```

3. Machine learning model & Preliminary results

I trained the dataset by creating a CNN model in KERAS.

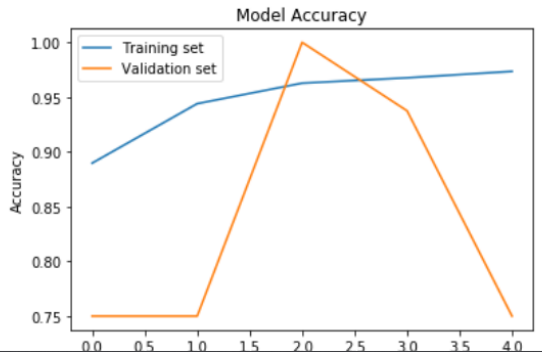
```
[9]  1 # Multilayer Perceptron (MLP) for multi-class softmax classification:
     2 # source: https://keras.io/getting-started/sequential-model-guide/
     3 model = Sequential()
     4 model.add(Dense(64, activation='relu', input_dim=20))
     5 model.add(Dropout(0.5))
     6 model.add(Dense(64, activation='relu'))
     7 model.add(Dropout(0.5))
     8 model.add(Dense(10, activation='softmax'))
     9
    10 sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
    11 model.compile(loss='categorical_crossentropy',
    12               optimizer=sgd,
    13               metrics=['accuracy'])
    14
    15 model.fit(x_train, y_train,
    16           epochs=20,
    17           batch_size=128)
    18
```

```
optimizer = Adam(lr = 0.0001)
early_stopping_monitor = EarlyStopping(patience = 3, monitor = "val_acc", mode="max", verbose = 2)
model.compile(loss="categorical_crossentropy", metrics=["accuracy"], optimizer=optimizer)
history = model.fit_generator(epochs=5, callbacks=[early_stopping_monitor], shuffle=True, validation_data=valid_batches, gener
prediction = model.predict_generator(generator=train_batches, verbose=2, steps=100)
```

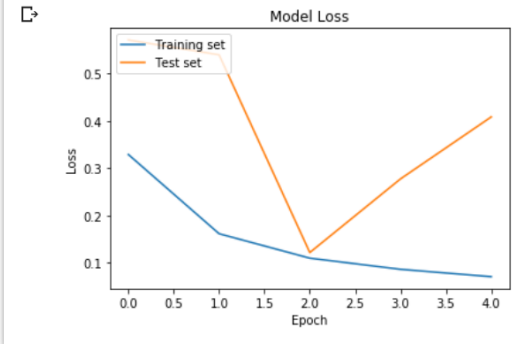
```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizers.py:793: The name tf.train.Optimizer is deprecated.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3576: The name tf.log is deprec

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/python/ops/math_grad.py:1424: where (from tensorflo
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:1033: The name tf.assign_add is

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:1020: The name tf.assign is dep

Epoch 1/5
 - 380s - loss: 0.3287 - acc: 0.8897 - val_loss: 0.5707 - val_acc: 0.7500
Epoch 2/5
 - 379s - loss: 0.1613 - acc: 0.9440 - val_loss: 0.5393 - val_acc: 0.7500
Epoch 3/5
 - 377s - loss: 0.1097 - acc: 0.9628 - val_loss: 0.1216 - val_acc: 1.0000
Epoch 4/5
 - 377s - loss: 0.0861 - acc: 0.9676 - val_loss: 0.2772 - val_acc: 0.9375
Epoch 5/5
 - 378s - loss: 0.0706 - acc: 0.9735 - val_loss: 0.4081 - val_acc: 0.7500
```

If the accuracy of the validation set is similar to the accuracy of the training set, my model is good. However, if it is much lower than the training set, it means my model is overfitting. So I plotted the graph of the model performance to improve the accuracy.

```
[13]  # Accuracy
      plt.plot(model.history.history['acc'])
      plt.plot(model.history.history['val_acc'])
      plt.title('Model Accuracy')
      plt.ylabel('Accuracy')
      plt.xlabel('Epoch')
      plt.legend(['Training set', 'Validation set'], loc='upper left')
      plt.show()
```



```
# Loss
plt.plot(model.history.history['loss'])
plt.plot(model.history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Training set', 'Test set'], loc='upper left')
plt.show()
```



4. Next steps

- code a webapp

- try to implement a better model