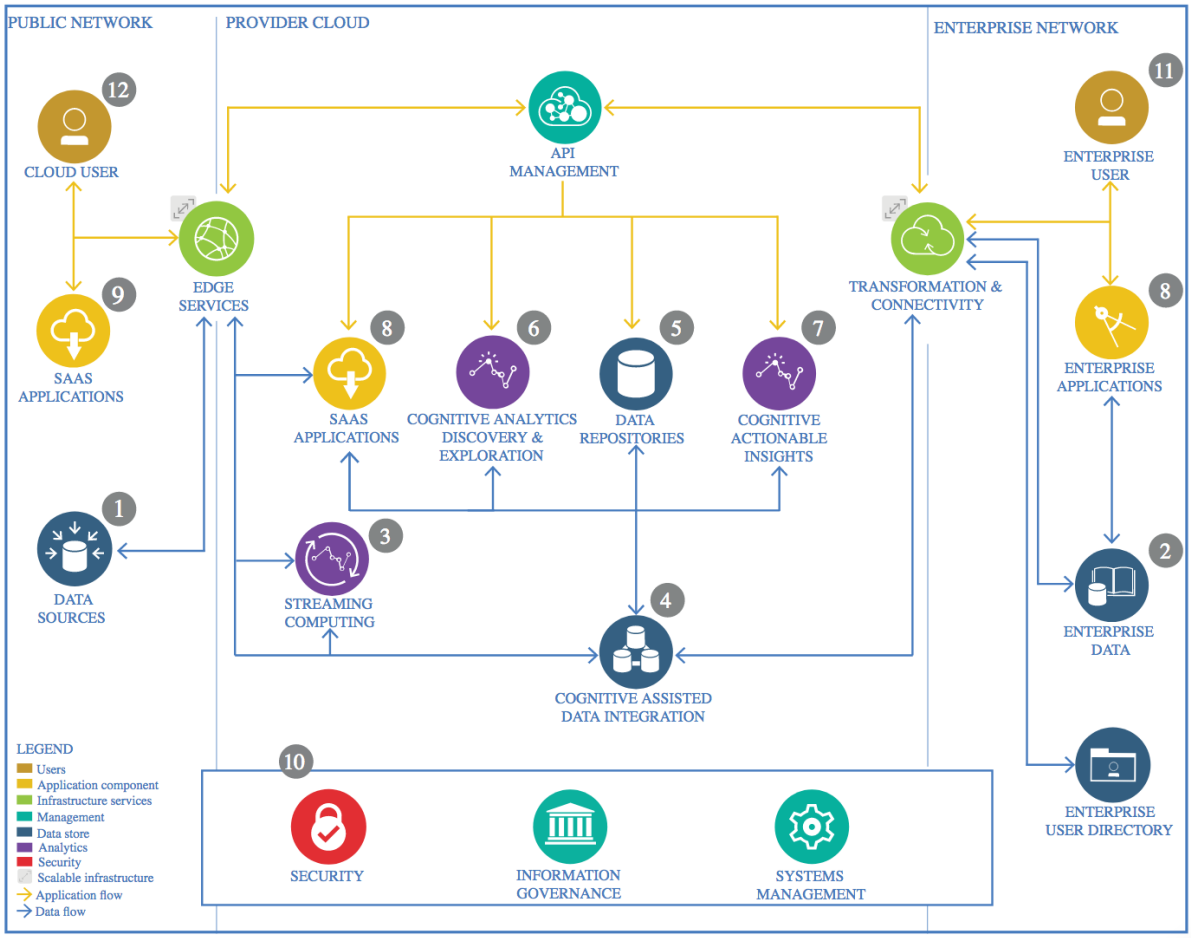
Pneumonia Detection ADD

By: Sergio Teso Lorenzo

1 Architectural Components Overview



IBM Data and Analytics Reference Architecture. Source: IBM Corporation

1.1 Data Source

1.1.1 Technology Choice

The data has been obtained from https://www.kaggle.com/. All input data are x-ray images of lungs. There are two types of classes for them: normal lungs and pneumonia ill lungs.

1.1.2 Justification

This dataset has been selected because it is a free and easy to use source of data and thanks to the user feedback it presents, we can have a degree of confidence in the quality of the data.

In an initial exploration of it we can see the amount and ratio of each class for the datasets that are going to be used.

```
Total Data: 5856
Train data: 5216
Test data: 624
Validation data: 16
```

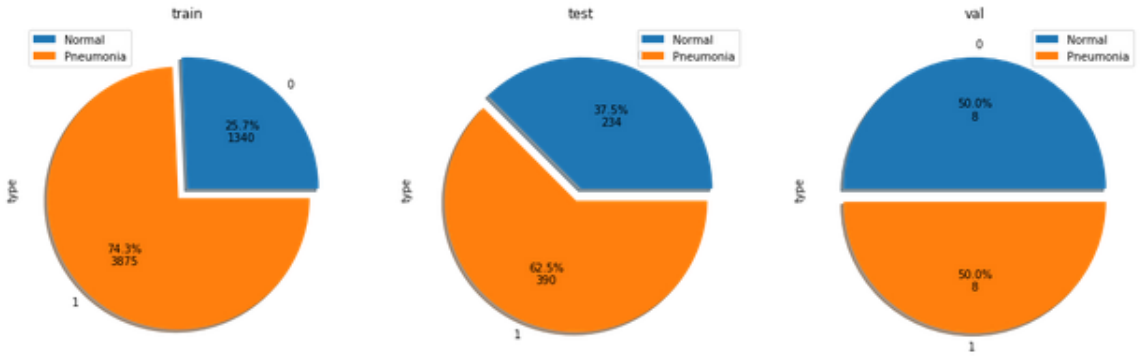


Fig 1: Initial data exploration

We can see that generally we have a higher percentage of Pneumonia class than normal class. This can lead to a better efficiency in classifying pneumonia lungs than normal ones, which corresponds with the best security we want; being able to correctly classify ill lungs as ill so we don't tell an ill person he is healthy.

1.2 Enterprise Data

1.2.1 Technology Choice

Component not needed.

1.2.2 Justification

This project uses a small amount of static data, around 1 GB so it is intended to run locally. Because of this, not cloud solutions are needed now.

1.3 Streaming analytics

1.3.1 Technology Choice

This model will only use batch-processing.

1.3.2 Justification

The database to use is a static dataset of images not bound to real-time events. So, it will only use batch processing. This process consists of loading the data in batches instead all at once, so the computer memory is used in a more efficient manner.

1.4 Data Integration

1.4.1 Technology Choice

Pandas library is used to import all the data and separate it in train, test, and validation subsets.
Keras image libraries are used to transform the input image data.

1.4.2 Justification

Because all the input data consists of images, we can apply image transformation techniques like rotations, shifts and zoom-ins to increase the amount of data without hurting the model performance. Also, because images are interpreted as integer number matrices, there are going to be normalized between 0 and 1.

1.5 Data Repository

1.5.1 Technology Choice

Local file system.

1.5.2 Justification

As stated before, due to the small size of the data it can be stored locally without requiring cloud-based solutions. Furthermore, this data was obtained from https://www.kaggle.com/ so there is no danger of losing the data.

1.6 Discovery and Exploration

1.6.1 Technology Choice

We have an x-ray images database of normal and pneumonia ill lungs, each with its class label assigned and organized in the train, test and validation subsets. The objective is to elaborate a deep-learning model using the Keras library that learns to classify an x-ray image of lungs as normal or pneumonia ill.

1.6.2 Justification

Keras Libraries are used due to being easy and free to implement while having great performance.

1.7 Actionable Insights

1.7.1 Technology Choice

Keras Convolutional Neural Network model.

1.7.2 Justification

Due to the dataset having only images as input data I have chosen to use a convolutional neural network (CNN). Because images consist of numeric values for each pixel, in a normal fully connected feed-forward network we would have a very large vector as input for each image which usually leads to overfitting. In the other hand, CNN's can use a two-dimensional filter in images that creates small subsets of neurons that try to identify a concrete feature in the image.

A good way of imagining this process is to think in a picture of a cat. Then you divide that picture in small squares, therefore in one you can only see one leg, the tail, one ear... etc. With this you have a small subset of neurons, each trying to classify a smaller picture. For example, the subset that corresponds to the tail will try to identify if it is a cat's tail or a dog's one.

Because CNN has this good architecture for image recognition has been the reason why I have chosen to use a 2d CNN to classify my x-ray images of lungs.
In my concrete model there is a convolutional layer that extracts the features from the image. Then I add a pooling layer that down samples each feature, which reduces the dimensionality for faster computations and reduces the impact of the least important elements of the feature. Next, I use again this combination of convolutional and pooling layer, so it uses the basic features extracted in the first layer to get more complex features from the image. Finally, I flatten the feature matrix extracted by the convolutional layer into a vector for using as input in a final, more traditional, fully connected feed-forward layer. This final computation layer will classify the features and output to a single binary neuron if the image corresponds to Normal Lungs = 0 or Pneumonia ill lungs = 1.

For the evaluating model performance, I will create the confusion matrix and extract the precision recall and F1-score. Then, I will create the ROC curve and calculate the AUC value of it.

With these measurements in mind I created the first model using an SVM Linear model, so we can see a model that will underperform in comparation to the justified CNN model.

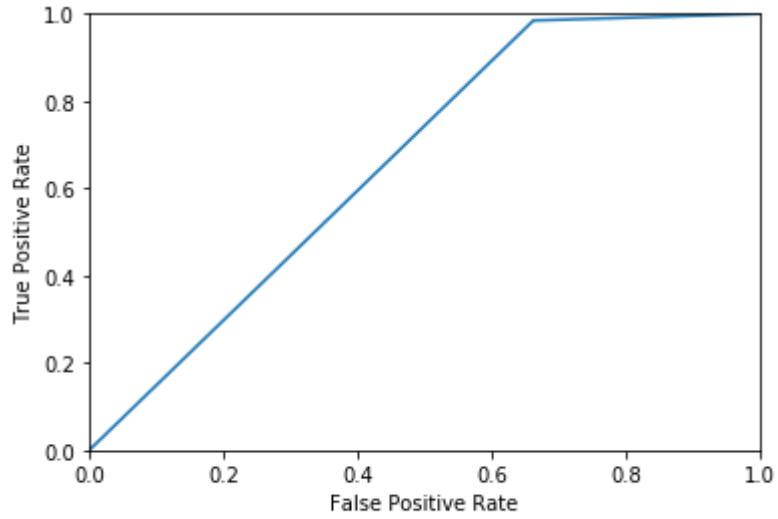
```
Confusion Matrix
[[ 79 155]
 [  6 384]]
Classification Report
              precision    recall  f1-score   support

   Normal(0)       0.93      0.34      0.50       234
Pneumonia(1)       0.71      0.98      0.83       390

    accuracy                           0.74       624
   macro avg       0.82      0.66      0.66       624
weighted avg       0.79      0.74      0.70       624
```

Fig3: SVM results

As seen in Fig3 we get a very good recall value for this pneumonia detection but poor for the normal class. SVM being a linear model and because our dataset has mostly pneumonia images over normal ones the model usually classifies input as pneumonia. So, we can't take much into account the good recall value for pneumonia and we will look to the AUC value for a better evaluation of the model performance.




```
Area under the curve: 0.6611111111111112
```

Fig4: SVM ROC curve and AUC value

As expected, the model performance is not very good because the AUC value of 0.66 is relatively small.

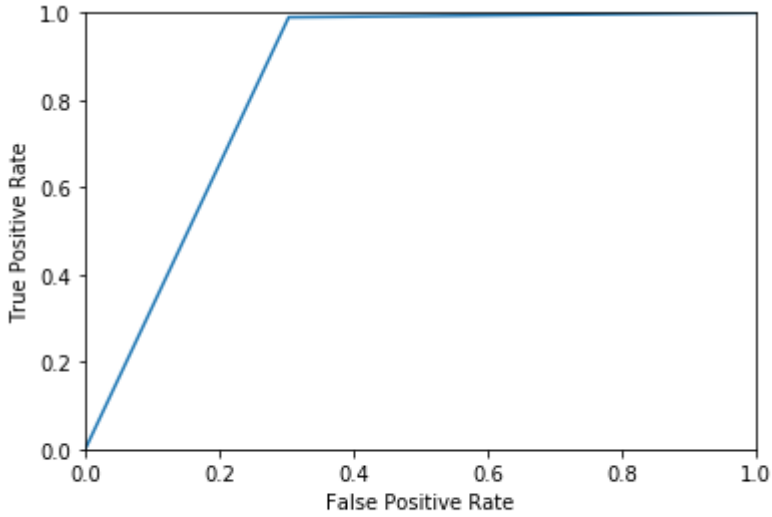
Next, we will see scores that the implementation of the Keras CNN archives.

```
Confusion Matrix
[[163  71]
 [  4 386]]
Classification Report
              precision    recall  f1-score   support

   Normal(0)       0.98      0.70      0.81       234
Pneumonia(1)       0.84      0.99      0.91       390

    accuracy                           0.88       624
   macro avg       0.91      0.84      0.86       624
weighted avg       0.89      0.88      0.87       624
```

Fig4: CNN model results.




```
Area under the curve: 0.8431623931623932
```

Fig5: CNN model ROC curve and AUC value.

Like it happened with the SVM model we get a good recall for the pneumonia class but a no so good recall for the normal class. In order to maximize the efficiency of the model in the real world we must try to even the F1-score, which is the harmonic mean of the precision and recall. For archiving it, I would change some hyperparameters of the CNN like the filters it uses or the kernel_size. Also, for preventing overfitting, I created an early stopping callback. It uses the small “validation_set” that gets classified after each training epoch. If the model gets above a 93% in accuracy and below 0.12 in the loss function for this 16-input set of 8 cases for each class, it stops the training for not overfitting the model. This were the best results archived:

```
model = load_model(filename)

#Confution Matrix and Classification Report
Y_pred = model.predict_generator(test_data_generation, test_steps, workers
# Get most likely class
y_pred = np.argmax(Y_pred, axis=1)

true_classes = test_data_generation.classes

print('Confusion Matrix')
print(confusion_matrix(true_classes, y_pred))
print('Classification Report')
target_names = ['Normal(0)', 'Pneumonia(1)']

report = classification_report(true_classes, y_pred, target_names=target_na
print(report)
```

```
Confusion Matrix
[[213  21]
 [ 36 354]]
Classification Report
              precision    recall  f1-score   support

   Normal(0)       0.86      0.91      0.88       234
Pneumonia(1)       0.94      0.91      0.93       390

    accuracy                           0.91       624
   macro avg       0.90      0.91      0.90       624
weighted avg       0.91      0.91      0.91       624
```

```
test_accu = model.evaluate_generator(test_data_generation,steps=validation_
print('The testing accuracy is :',test_accu[1]*100, '%')
```

```
The testing accuracy is : 96.875 %
```

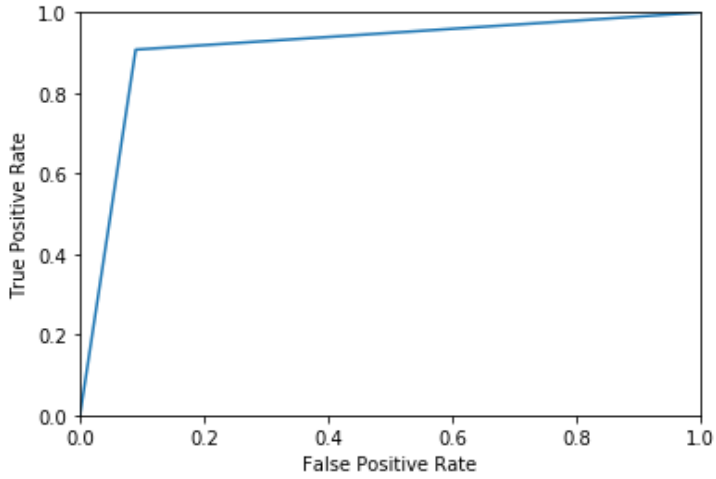
Fig6: Results obtained after training and evaluating the best model.

As seen in Fig6, we can observe the good results of our model. It gets around a 97% of accuracy in the test dataset which the model hasn't seen before. In the confusion matrix it can be appreciated that the model has more ease classifying correctly pneumonia lungs rather normal ones, but it maintains a similar recall and F1-score, so it is classifying smarter than previous models.

```
# Get true positives rate (TPR = recall) and false positive rate (FPR)
fpr , tpr , thresholds = roc_curve(true_classes, y_pred)

plt.plot(fpr,tpr)
plt.axis([0,1,0,1])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()

auc_score=roc_auc_score(true_classes, y_pred)
print("Area under the curve: {}".format(auc_score))
```




```
Area under the curve: 0.9089743589743589
```

Fig7: ROC curve and AUC value of the best model

Finally, in Fig7 we can see the ROC curve of the model which tell us how much the model is able to distinguish between classes. The AUC (area under the curve) is a value that the nearer it is from 1 the better. In this case we have a model that archived an AUC of 0.90 which means it does a good job distinguishing and correctly classifying each class. Because it is the higher value of the previous models, we can assure it is the best model we have created which is sustained by the testing accuracy it got.

1.8 Applications / Data Products

1.8.1 Technology Choice

The data product is a jupyter notebook.

1.8.2 Justification

The reason for developing a jupyter notebook is due to easy access it provides to be read, modified and ran.

1.9 Security, Information Governance and Systems Management

1.9.1 Technology Choice

The source code will be open source to anyone that wants to use it or improve it. The dataset is open source provided by https://www.kaggle.com/.

1.9.2 Justification

Because the project uses an open source dataset and it has been developed for academic use, it will be open source. This can help other data-scientist in their career development and goes along the open-source guidelines.