AUTOMATIC DIAGNOSIS OF RESPIRATORY DISORDERS

with Audio Analysis

RÉSUMÉ

Respiratory disorders are part of the 21th century global health diseases. What if automatic diagnosis was available for everyone, anywhere?

Bruno Nassivet

Capstone Project for the IBM Advanced Data Science Professional Certificate

Table of content

Table of figures

1 Introduction

Respiratory diseases like Asthma[1] and Chronic Obstructive Pulmonary Disease (COPD) are developing all over the world. All health organizations in the world, either national or international like the World Health Organization, are leading actions to detect, prevent, diagnose

As an example, currently more than 25 million people in the United States have asthma. Approximately 14.8 million adults have been diagnosed with COPD, and approximately 12 million people have not yet been diagnosed.

Moreover, significant disparities in asthma morbidity and mortality exist, particularly for low-income and minority populations, and other factors like geographic isolation is an significant factor of death due to respiratory diseases. Degraded air quality is another factor of respiratory diseases.

What if respiratory diseases could be detected on the first symptoms without any logistic burden?

What if the identification and diagnosis of respiratory diseases could be done anywhere, at any time?

My capstone project consists on automatic diagnosis of respiratory disease by analysis and classification of respiration audio recordings and associated diagnosis meta data.

Respiratory sounds are important indicators of respiratory health and respiratory disorders. Automated machine-aided pre-diagnosis can help accelerate patient handling via remote or automated disease detection and classification.

2 Architectural Components Overview

This project has been realized following the Lightweight IBM Cloud Garage Method for Data Science

This report document is structured according to the IBM Data and Analytics Reference Architecture - Architectural Decisions Document Template.

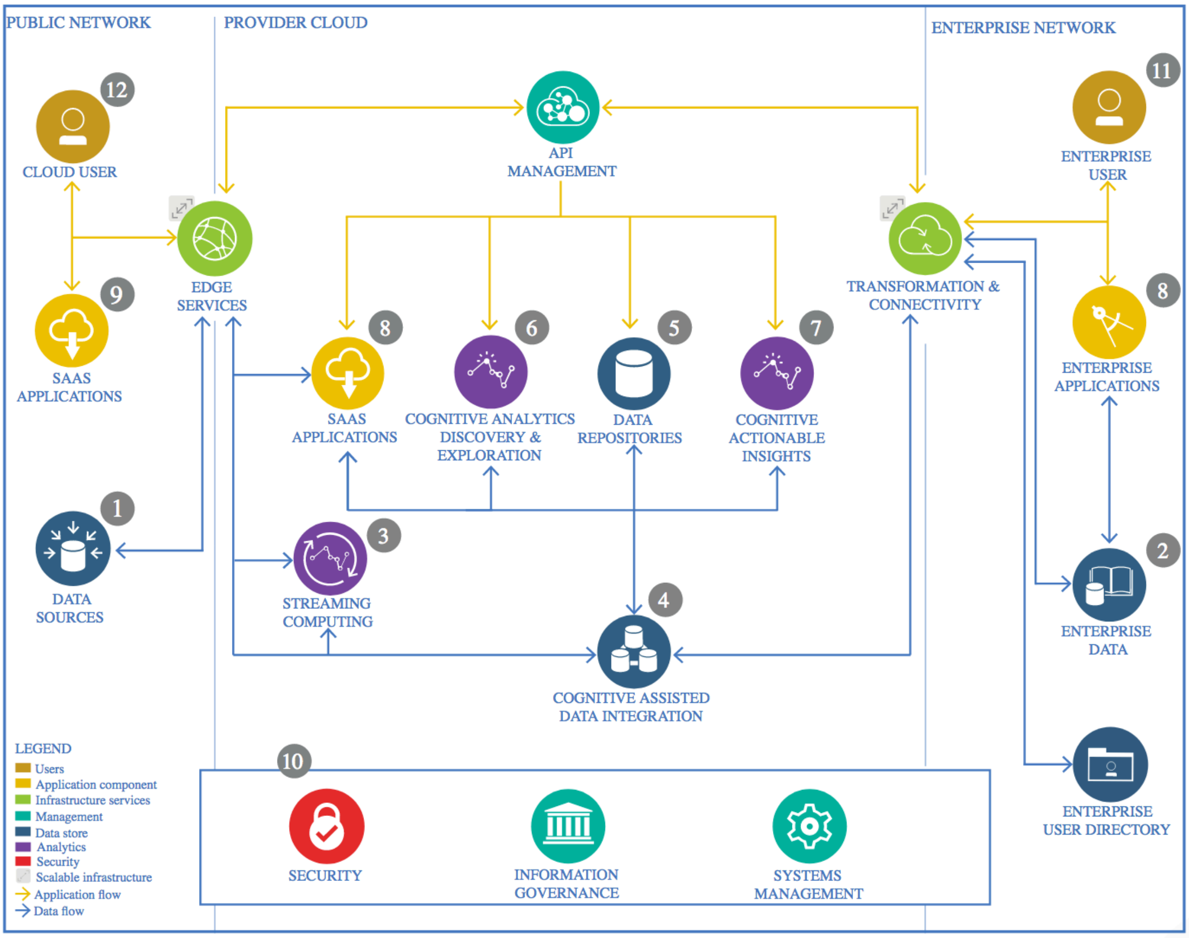


Figure 1 - IBM Data and Analytics Reference Architecture. Source: IBM Corporation

2.1 Data Source

2.1.1 Technology Choice

The data source used for this step of the project is the Respiratory Disorder Analysis dataset, a static Kaggle dataset available at: https://www.kaggle.com/vbookshelf/respiratory-sound-database/home .

The Respiratory Sound Database was created by two research teams in Portugal and Greece. It includes 920 annotated recordings of varying length - 10s to 90s. These recordings were taken from 126 patients. There are a total of 5.5 hours of recordings containing 6898 respiratory cycles - 1864 contain crackles, 886 contain wheezes and 506 contain both crackles and wheezes. The data includes both clean respiratory sounds as well as noisy recordings that simulate real life conditions. The patients span all age groups - children, adults and the elderly.

This Kaggle dataset includes:

- 'patient_diagnosis.csv: a .csv text file listing the diagnosis for each patient
- demographic_info.txt: a text file containing demographic information for each patient (note: the first line is blank and is deleted manually)
- audio_and_txt_files : folder containing
    - 920 .wav sound files of respiratory audio recording
    - 920 annotation .txt files

The demographic info file has 6 columns:

1. Patient number
2. Age
3. Sex
4. Adult BMI (kg/m2)
5. Child Weight (kg)
6. Child Height (cm)

Each audio file name is divided into 5 elements, separated with underscores (_).

1. Patient number (101,102,...,226)
2. Recording index
3. Chest location
    a. Trachea (Tc)
    b. Anterior left (Al)
    c. Anterior right (Ar)
    d. Posterior left (Pl)
    e. Posterior right (Pr)
    f. Lateral left (Ll)
    g. Lateral right (Lr)
4. Acquisition mode
    a. sequential/single channel (sc),
    b. simultaneous/multichannel (mc)
5. Recording equipment
    a. AKG C417L Microphone (AKGC417L),
    b. 3M Littmann Classic II SE Stethoscope (LittC2SE),
    c. 3M Litmmann 3200 Electronic Stethoscope (Litt3200),
    d. WelchAllyn Meditron Master Elite Electronic Stethoscope (Meditron)

The annotation text files have four columns:

- Beginning of respiratory cycle(s)
- End of respiratory cycle(s)
- Presence/absence of crackles (presence=1, absence=0)
- Presence/absence of wheezes (presence=1, absence=0)

The abbreviations used in the diagnosis file are:

- COPD: Chronic Obstructive Pulmonary Disease
- LRTI: Lower Respiratory Tract Infection
- URTI: Upper Respiratory Tract Infection

The annotation files is looked at but is not used in the analysis, since they are made available from a specific analysis, while the goal is to diagnosis from measured data.
At this step of the project, which is a preliminary Proof of Concept (PoC), no Data Source import technology have been defined.

2.1.2 Justification

The data source used for this step of the project is the Respiratory Disorder Analysis dataset, a static Kaggle data source consisting of files. The dataset is to be downloaded as a zip file, and unzipped into the selected data repository for processing.

2.2 Enterprise Data

2.2.1 Technology Choice

At this step of the project, no Enterprise Data technology have been defined.

2.2.2 Justification

At this step, the data imported from the Data Source are the only data necessary for the project.

2.3 Streaming analytics

2.3.1 Technology Choice

At this step of the project, no Streaming Analytics technology have been defined.

2.3.2 Justification

At this step of the project, deployment is not considered. The first step consists on a preliminary analysis of the data, models and processing, with a batch processing approach. The outcomes of this first steps are necessary to define the streaming analytics technologies adequate to the use case, in term of deployment and non-functional requirements such as throughput or latency . Streaming, from Edge measurement to data integration, will be addressed on the next iterations of the project.

2.4 Data Integration

2.4.1 Technology Choice

The data integration and feature extraction is implemented in two notebooks:

- ***'BNA_Capstone_Feature_Extraction.ipynb'***: in this notebook, the full audio recording files are resampled at the same rate and split in small chunks of the same size; MFCC transformation is then run on all samples, and the features dataset in saved on disk in json format. Given the input files, this lead to an unbalanced dataset with a large majority of samples corresponding to a diagnosis COPD.

- ***'BNA_Capstone_Feature_Extraction-with_Data_Augmentation_v1.2.ipynb'***: the goal of this notebook is to generate a more balanced features dataset, with number of sample per diagnosis type approximatively at TARGET_SAMPLE=1000. To achieve this goal, a sample of records is randomly picked up for diagnosis types greater than TARGET_SAMPLE, while data augmentation is performed on the dataset for diagnosis types with a number of records less than TARGET_SAMPLE, applying signal alteration:

- white noise addition,
- stretching.

In order not to affect the models, a subset of samples is kept out of the data augmentation processes: these are targeted to constitute the test set for the models to be evaluated. The resulting dataset is saved on disk in json format.

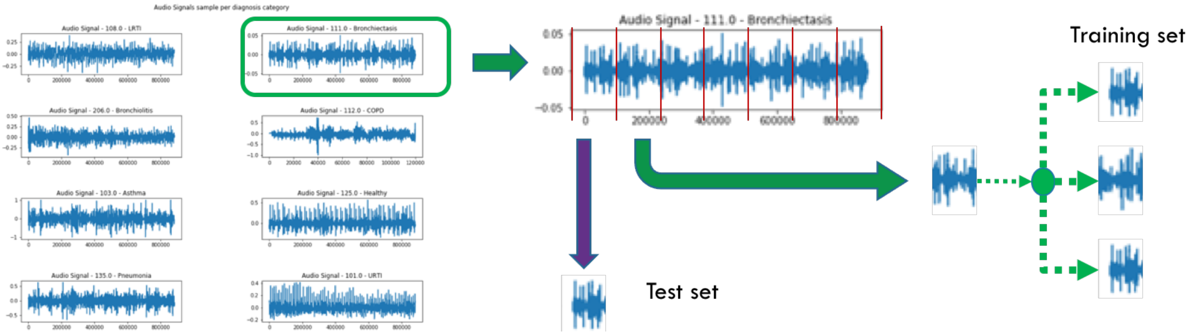

Audio Signals sample per diagnosis category
Audio Signal - 108.0 - LRTI
Audio Signal - 111.0 - Bronchiectasis
Audio Signal - 206.0 - Bronchiolitis
Audio Signal - 112.0 - COPD
Audio Signal - 103.0 - Asthma
Audio Signal - 125.0 - Healthy
Audio Signal - 135.0 - Pneumonia
Audio Signal - 101.0 - URTI
Audio Signal - 111.0 - Bronchiectasis
Training set
Test set

Figure 2 - Data augmentation processing

Data integration is mainly implemented using Apache Spark along with Apache SparkSQL. Pandas dataframes and Scikit-learn are also used as part of this first iteration.
librosa and numpy are used for the generation of the MFCC features. The dataset output from the feature generation is saved on disk to be further made available to models.
In summary, the outputs from the feature extraction steps from the input Respiratory Disorder dataset are synthetized on the following table.

Output Name	Output description	Notebook
respiratoryData_12kbps	Folder containing the json export of the dataset feature for all audio files splitted, resampled at 12Kbps, transformed to MFCC coefficients, with columns : - filename - patientId - recIndex - diagnosisIndex - mfcc coefficients	BNA_Capstone_Feature_Extraction.ipynb
respiratoryDataAugmented_v1.2	Folder containing the json export of the dataset feature for audio files splitted, resampled at 12Kbps, with data augmentation of sub sample, transformed to MFCC coefficients, with columns : - filename - patientId - recIndex - diagnosisIndex - mfcc coefficients	BNA_Capstone_Feature_Extraction-with_Data_Augmentation_v1.2.ipynb

Output Name	Output description	Notebook
	- recSource indicator - with augmentation or not	

2.4.2 Justification

The Respiratory disorder dataset is constituted of a mix of tabular data and .wav audio data representing more than 2 GB on disk . For this primary step of the project, the objective is to be able to execute all the processing on a simple MacBook (Core I5 processor, 8GB RAM, 40GB of available storage). While the computing and storage power for ETL processing is relatively limited for this first step of the project, much more significant resources will be required when deploying at scale.

Apache Spark choice is driven by:

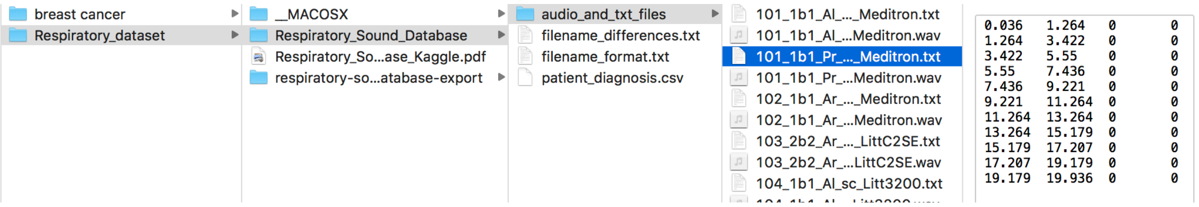
- Its ability to parallelize storage and processing on the dataset through Datasets and ML pipelines,
- Its Scalability, with the ability to run either on a simple computer or on a cluster.
- Its capability to handle both batch and stream processing, the latter being envisioned for future deployment.

Pandas dataframes are mainly used to alleviate the performance constraints on the standalone deployment for the first step.

The glob library is used for directories and files I/O.

2.5 Data Repository

2.5.1 Technology Choice



breast cancer
Respiratory_dataset

__MACOSX
Respiratory_Sound_Database
Respiratory_So...ase_Kaggle.pdf
respiratory-so...atabase-export

audio_and_txt_files
filename_differences.txt
filename_format.txt
patient_diagnosis.csv

101_1b1_Al_..._Meditron.txt
101_1b1_Al_...Meditron.wav
101_1b1_Pr_..._Meditron.txt
101_1b1_Pr_...Meditron.wav
102_1b1_Ar_..._Meditron.txt
102_1b1_Ar_...Meditron.wav
103_2b2_Ar_..._LittC2SE.txt
103_2b2_Ar_...LittC2SE.wav
104_1b1_Al_sc_Litt3200.txt

```
0.036   1.264   0       0
1.264   3.422   0       0
3.422   5.55    0       0
5.55    7.436   0       0
7.436   9.221   0       0
9.221   11.264  0       0
11.264  13.264  0       0
13.264  15.179  0       0
15.179  17.207  0       0
17.207  19.179  0       0
19.179  19.936  0       0
```

Selected data of the input dataset are stored on a selected folder on the OS file system.

2.5.2 Justification

The dataset size is around 2GB and is composed of regular files, which does not need specific repository technology. It also provide a simple universal deployment on the target laptop, but also on cloud storages.

The data repository strategies will have to be reconsidered when running at scale.

2.6 Discovery and Exploration

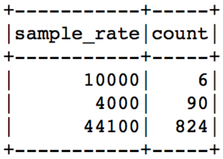
2.6.1 Technology Choice

The data discovery and exploration path is supported by the notebook "BNA_Capstone_DataAnalysis_nb.ipynb". The following technologies are used for discovery and exploration:

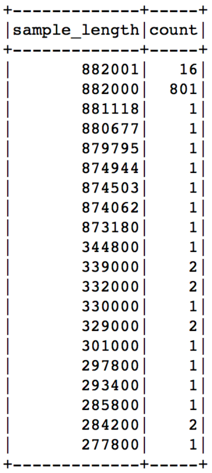
- Jupyter notebook,
- Python 3.6.8,
- Pyspark 2.4.1, including mllib for statistics,
- Numpy 1.16.2,
- scikit-learn 0.19.1,
- pandas 0.18.1,
- matplotlib 2.1.0
- For audio file processing:
  - Soundfile 0.10.2: read .wav file; https://pysoundfile.readthedocs.io/en/0.9.0/
  - librosa 0.6.3: audio signal manipulation (e.g. resampling, normalization) and display; https://librosa.github.io/librosa/

The preliminary step of data discovery and exploration consists of loading and assembling into one Spark dataframe the different information contained on the Respiratory Disorder dataset from the different files, through parsing ang SparkSQL aggregations and joins.

In the second part, data discovery, the different distribution of data are highlighted.
It appears that a major part of information related to demography and recording equipment are sparse, with a lot of fields with value missing.
Regarding audio samples, there is a variety of recording length and sample rates, with a majority of long samples with high rates.

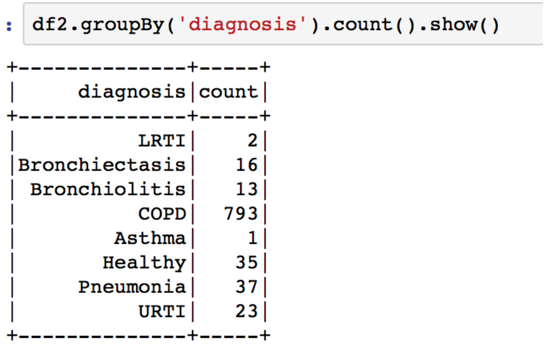


```
+-----------+-----+
|sample_rate|count|
+-----------+-----+
|      10000|    6|
|       4000|   90|
|      44100|  824|
+-----------+-----+
```



```
+-------------+-----+
|sample_length|count|
+-------------+-----+
|       882001|   16|
|       882000|  801|
|       881118|    1|
|       880677|    1|
|       879795|    1|
|       874944|    1|
|       874503|    1|
|       874062|    1|
|       873180|    1|
|       344800|    1|
|       339000|    2|
|       332000|    2|
|       330000|    1|
|       329000|    2|
|       301000|    1|
|       297800|    1|
|       293400|    1|
|       285800|    1|
|       284200|    2|
|       277800|    1|
+-------------+-----+
```

It is also highlighted that the dataset is unbalanced regarding the types of diagnosis: for 920 diagnoses, 793 are COPD, representing 88% of the diagnoses.



```
: df2.groupBy('diagnosis').count().show()
```

```
+--------------+-----+
|     diagnosis|count|
+--------------+-----+
|          LRTI|    2|
|Bronchiectasis|   16|
| Bronchiolitis|   13|
|          COPD|  793|
|        Asthma|    1|
|       Healthy|   35|
|     Pneumonia|   37|
|          URTI|   23|
+--------------+-----+
```

On the third part of the data analysis, we attempt to determine if some of the demographic data (sex, weight, …) are of influencing the diagnosis. For this, a test of independence using both correlation and Chi Square tests are evaluated, considering the following features: `'diagnosisIndex', 'age', 'sexIndex', 'adultBMI', 'sample_rate', 'sample_length', 'chLocIndex', 'acqModeIndex', 'recEqptIndex'.`
The tests show no dependency between diagnosis and other features.

Since the initial goal of the project is to auto diagnose a patient based patient information and raw recorded audio data, the annotation files, where crackles and wheezes are tagged per respiratory cycles, are not taken into account for the remaining of the analysis.

The last step of the dataset discovery and exploration consists of a visual analysis of the recorded audio data.
A sample with one record for each diagnosis is extracted from the audio samples.

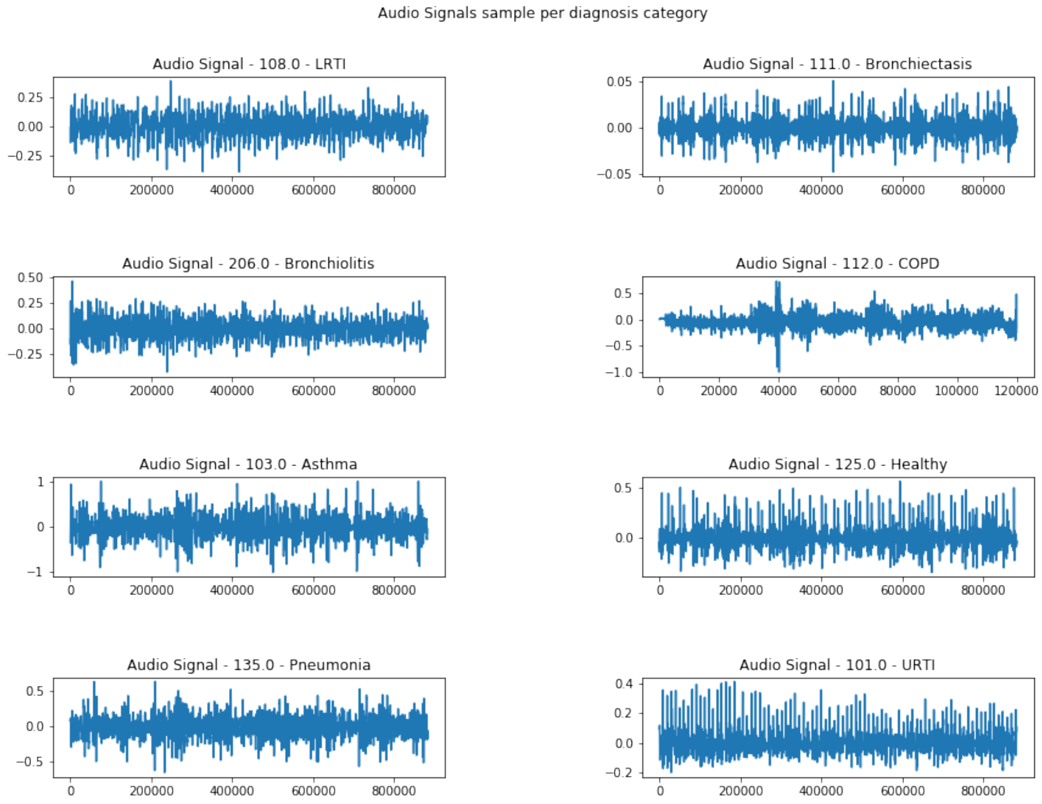


Figure 3- Sample audio signals per diagnosis category

The displays of audio signal as time series, along with spectrograms of the same samples show specific signatures for the different respiratory diseases picked-up in the sample, except LRTI and URTI which are very similar, and Pneumonia which is visually (on the inspected sample) close to the 2.

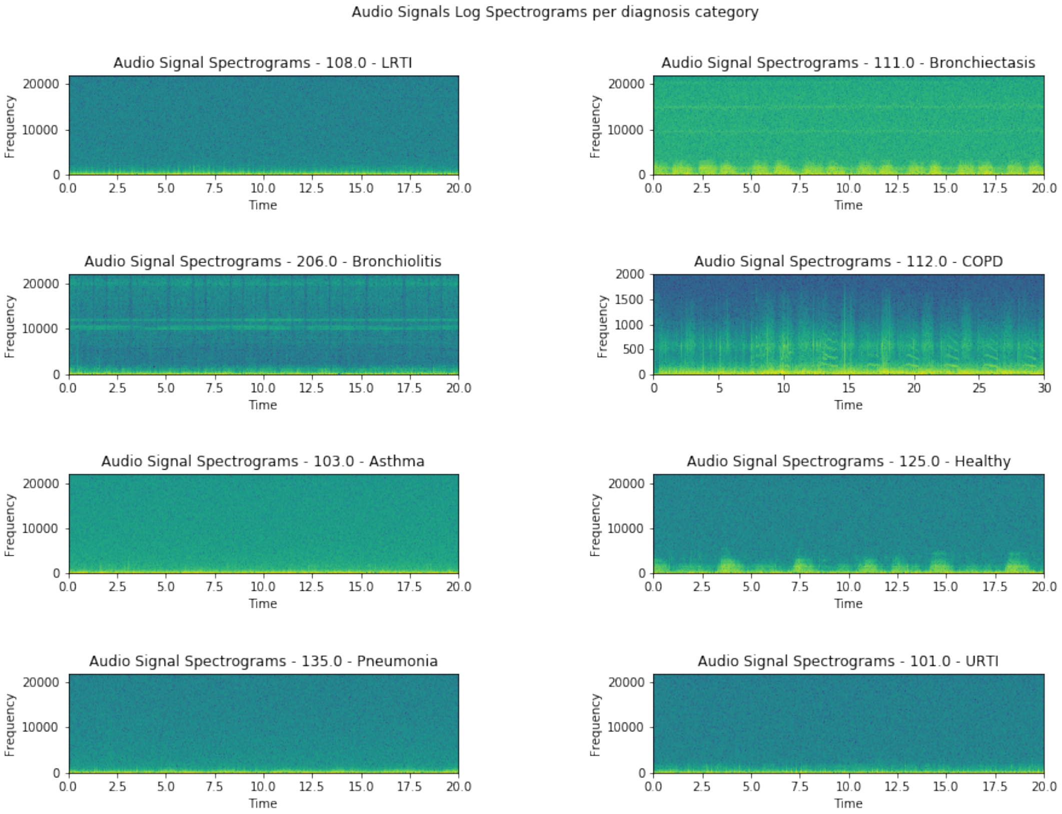


Figure 4 - Log Spectrograms sample per diagnosis type

This is even clearer when looking at spectrograms with a logarithmic scale, as below.

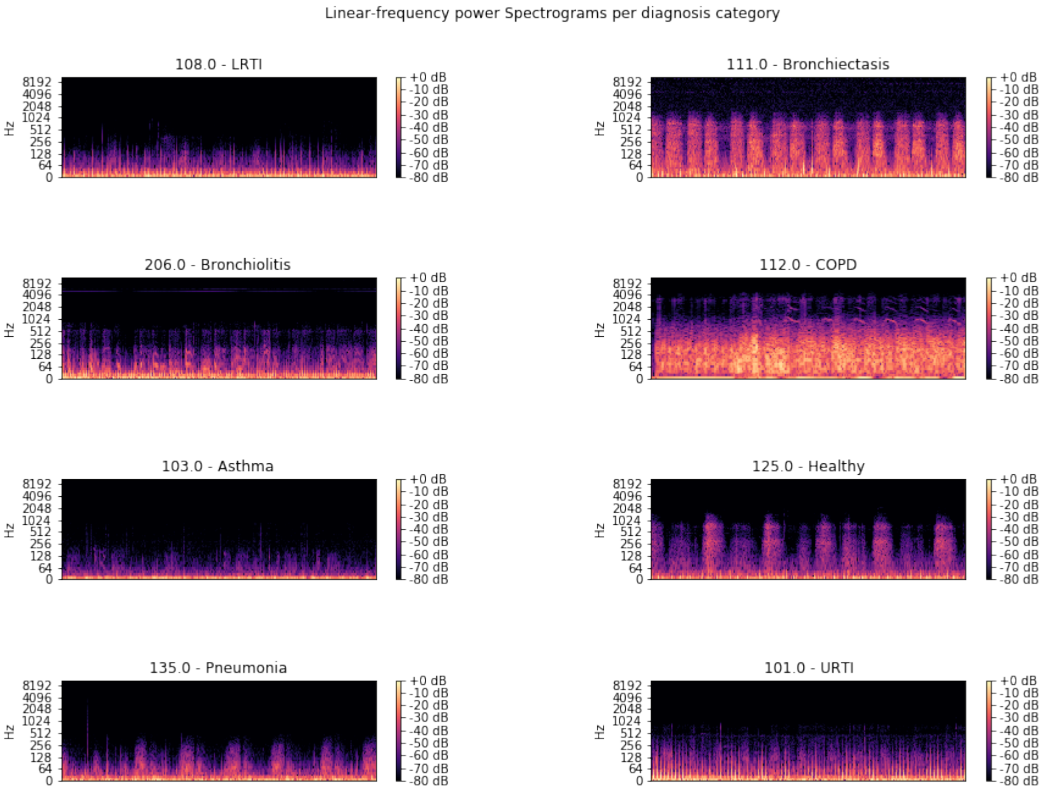


Figure 5 – Linear-frequency power spectrograms sample per diagnosis category

Finally, since records need to be normalized, the log spectrograms are displayed within 2 different sample rates: original 44100Hz and a reduced rate of 12000Hz.

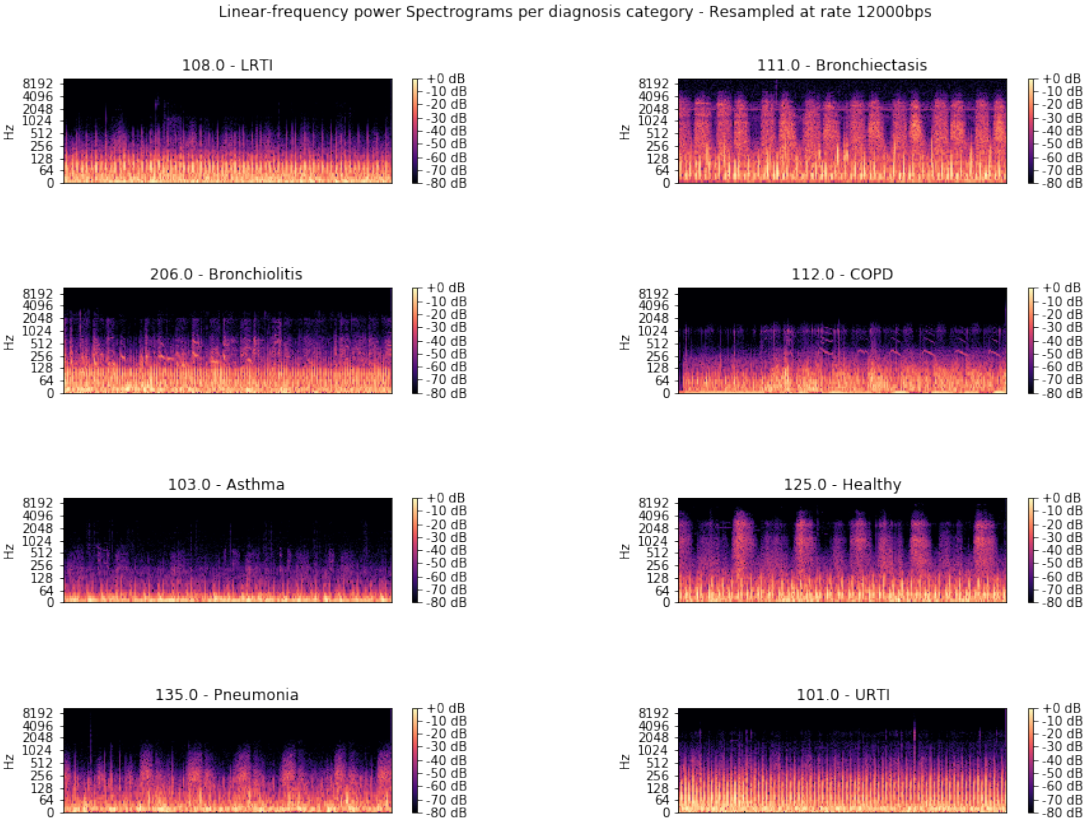


Figure 6 - Linear-frequency power spectrograms sample per diagnosis category - resamples at rate 12000bps

Even if the 12000Hz diagrams are less clear, they still allow to distinguish specific forms for the selected samples.
For the remaining of the project a sampling rate of 12000Hz will be used.

2.6.2 Justification

Mel-Frequency Cepstral Coefficients - MFCC – audio signal transformation choice is justified by its compact format compared to spectrogram or wavelet approaches.

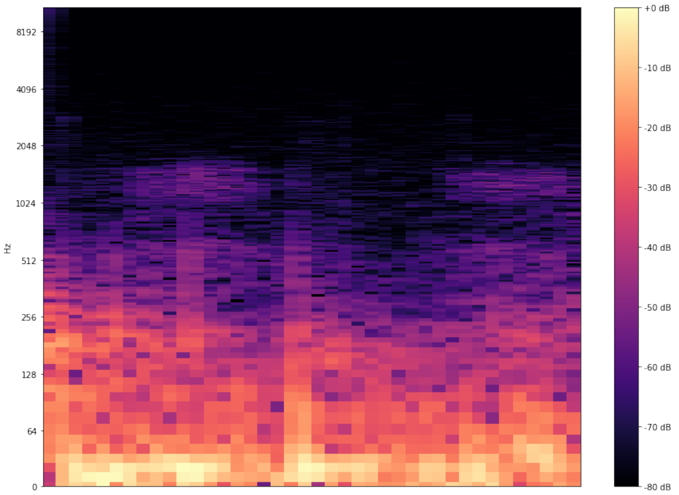


Figure 7 - Mel-Frequency Cepstral Coefficients - MFCC

The selected technologies are open source and widely supported, including on laptop and on IBM Watson Studio.
In particular:

- Jupyter notebook allows to easily share the project;
- Pandas has nicer table displays than Spark;
- matplotlib is versatile and sufficient for the displays needed for the project;
- librosa is a rich and widely used audio signal processing library, and includes various signal processing techniques along with built-in display (based on matplotlib). It supports the MFCC transformation algorithm, selected to be used for feature transformation of the audio signals, and has good processing performances.

The resampling at a sample rate of 12000Hz will be used, as a trade-off between model quality and memory consumption for performances purposes, and also in order to keep the 4000Hz samples without too much oversampling.

2.7 Actionable Insights

2.7.1 Technology Choice

2.7.1.1 Models

In order to perform multi-class classification of the Respiratory disorder features, four types of classifier models are defined and evaluated against the two generated data features (without and with data augmentation):

- Machine Learning models
  - Random Forest
  - Support Vector Machine (SVM)
- DL Models (on augmented data only)
  - Deep Feed Forward Neural Network (DFF-NN)
  - Convolutional Neural Network (CNN)

The Machine Learning models were first defined. The models performance evaluation show that the models performances on training set and prediction on the test set are very influenced by the unbalanced characteristic of the features, the records being mainly corresponding to diagnosis type COPD. This led to the generation of the balanced set of features with data augmentation.

	Dataset ➔	respiratoryData	respiratoryData_augmented
Notebook	Model		
BNA_Capstone_Model_ML-v1.0.1, BNA_Capstone_Model_ML-v1.3	Random Forest	x	x
BNA_Capstone_Model_ML-v1.0.1, BNA_Capstone_Model_ML-v1.3	SVM(kernel='poly', degree=8)	x	x
BNA_Capstone_Model_DL-v1.3	DFF-NN		x
BNA_Capstone_Model_DL-v1.3	CNN		x
BNA_Capstone_Model_DL-v1.3	CNN(with LeakyReLU layers)		x

The models parameters are available on the notebooks.

2.7.1.2 Models Performance Evaluation

Since this is a multi-class supervised classification problem, the metrics used for model evaluation are:

- precision,
- recall,
- F1-score,
- confusion matrix.

The models assessment is also performed regarding their generalization performance (underfitting/overfitting) by looking at the accuracy and loss for both training/validation and test sets.

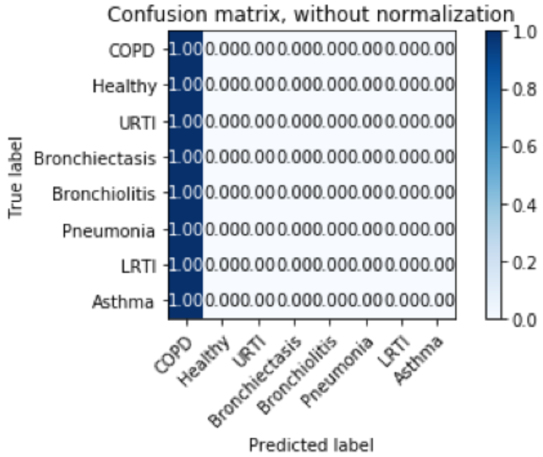
For DL neural network models, the drawing of the accuracy and loss curves (on Tensorboard and on Keras/Tensorflow) for both training and validation phases, provides a visual understanding of the learning curves.

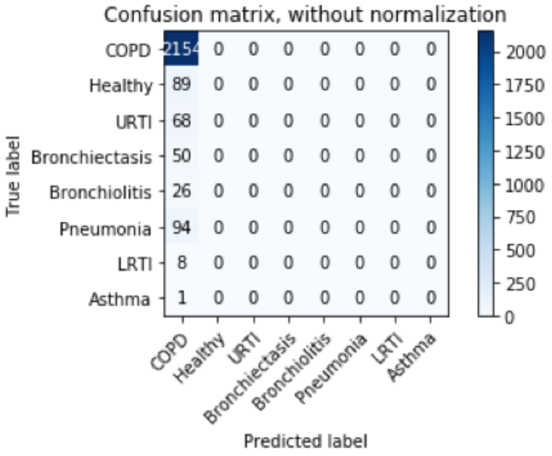
2.7.1.3 Results

2.7.1.3.1 First Iteration on Respiratory data (with no augmentation)

The first iteration was conducted on the respiratory dataset.
The dataset is split into 70% of training data and 30% of test data.

Random Forest





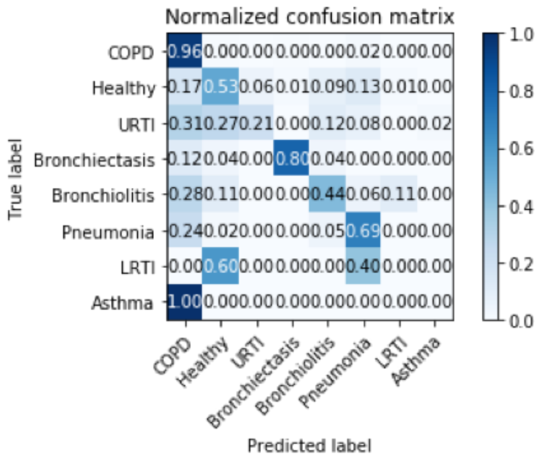
The Random Forest model could not categorize any data, and predicts only the most represented diagnosis type, COPD.

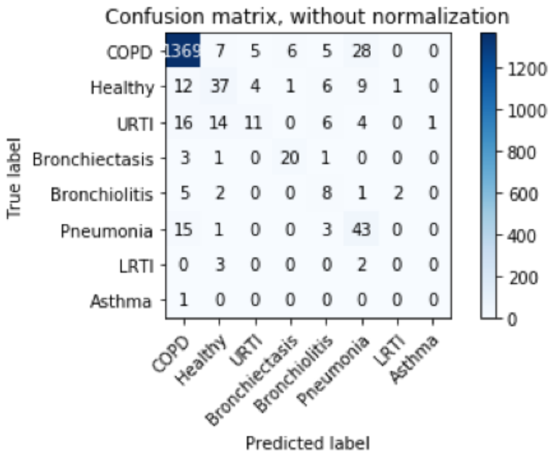
SVM

SVM gives the following prediction performances.

```
             precision    recall  f1-score   support

        0.0       0.96      0.96      0.96      1420
        1.0       0.57      0.53      0.55        70
        2.0       0.55      0.21      0.31        52
        3.0       0.74      0.80      0.77        25
        4.0       0.28      0.44      0.34        18
        5.0       0.49      0.69      0.58        62
        6.0       0.00      0.00      0.00         5
        7.0       0.00      0.00      0.00         1

avg / total       0.90      0.90      0.90      1653
```

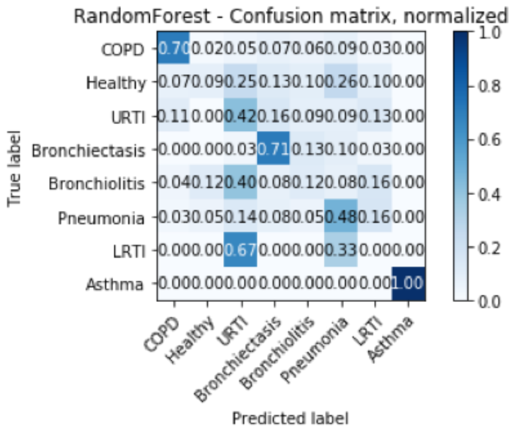
The results on the table are led by the unbalanced property of the dataset.
The confusion matrix below shows that even if SVM succeed in classifying some of the samples with lower occurrence, the overall result is driven by the COPD over-representation.

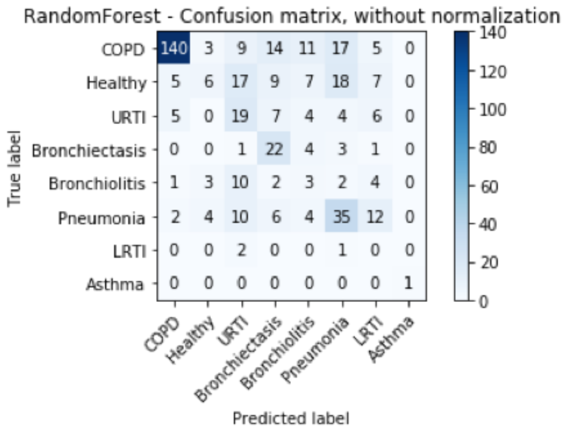




2.7.1.3.2 Second iteration: respiratory dataset with augmented data

2.7.1.3.2.1 Random Forest





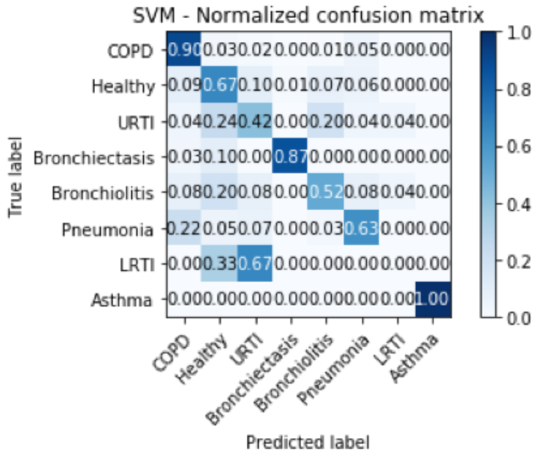
Even if better than in the first iteration, Random Forest classifier provides poor results.

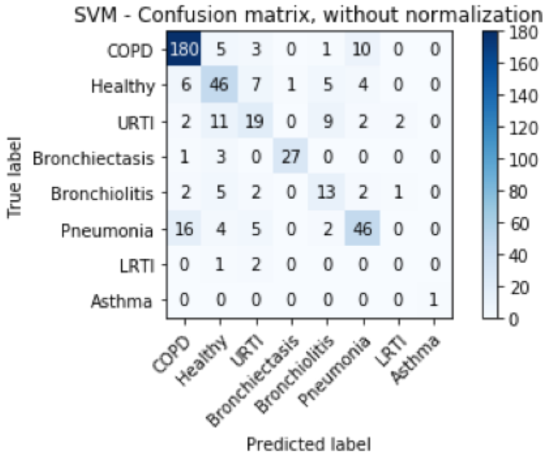
2.7.1.3.2.2 SVM

SVM with the same parameters gives the following prediction performances.

```
             precision    recall  f1-score   support

        0.0       0.87      0.90      0.89       199
        1.0       0.61      0.67      0.64        69
        2.0       0.50      0.42      0.46        45
        3.0       0.96      0.87      0.92        31
        4.0       0.43      0.52      0.47        25
        5.0       0.72      0.63      0.67        73
        6.0       0.00      0.00      0.00         3
        7.0       1.00      1.00      1.00         1

avg / total       0.74      0.74      0.74       446
```

The prediction performances are significantly increased.





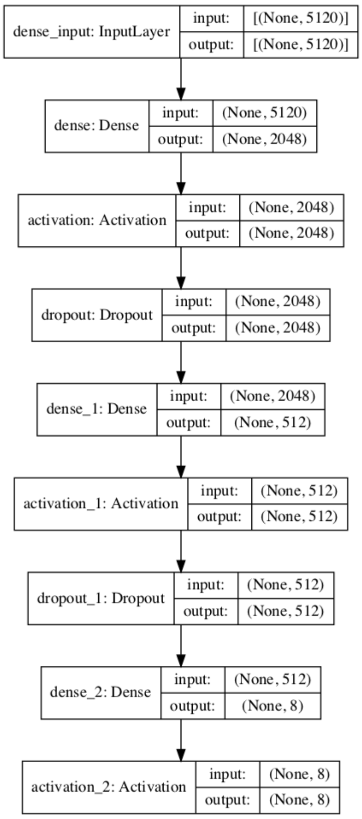
2.7.1.3.2.3 Deep Learning models

Several Deep Learning models were experimented. Their description and results are dumped on the paragraphs below:

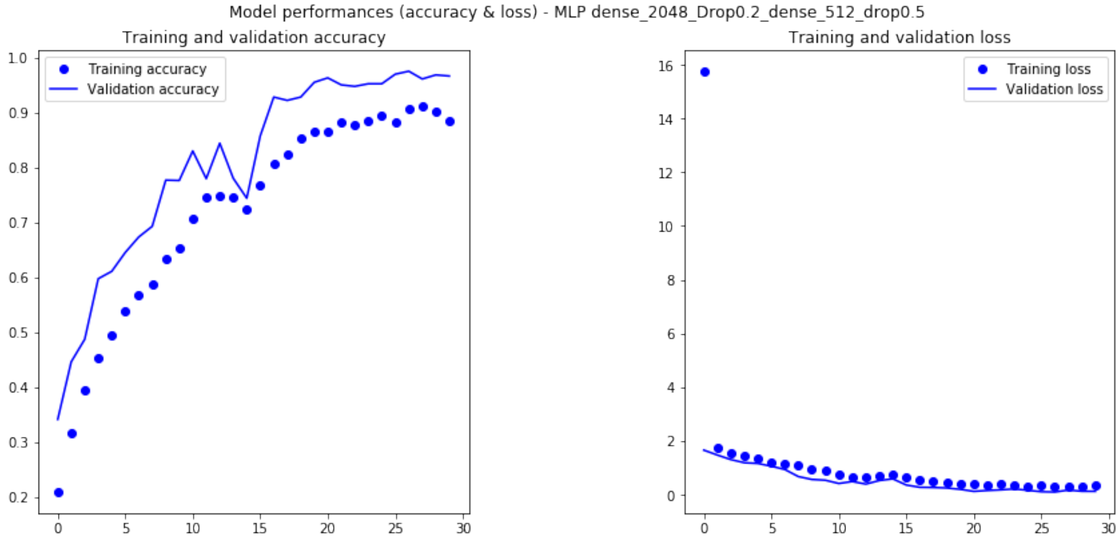
- Definition
- Model performance:
  - Training and validation accuracy and loss
  - Prediction performances:
    - Precision, recall, F1-score table,
    - Confusion matrix, normalized (between 0 and 1) or absolute.

These chapters are inserted for details, and can be skipped up to the Synthesis chapter.

2.7.1.3.2.4 Deep Feed Forward Neural Network (DFF-NN)

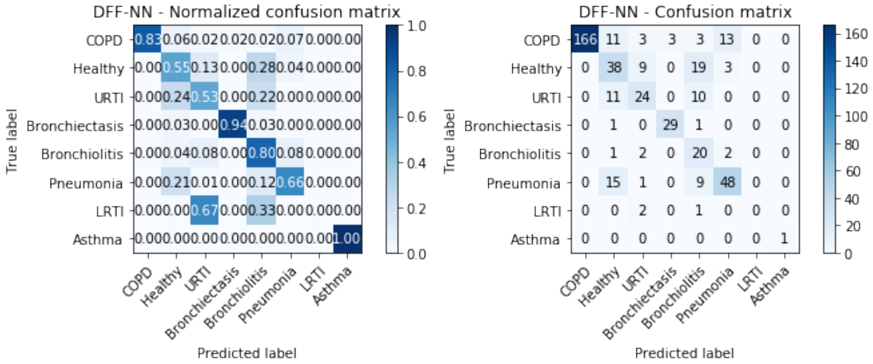


```
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 2048)              10487808
_________________________________________________________________
activation (Activation)      (None, 2048)              0
_________________________________________________________________
dropout (Dropout)            (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               1049088
_________________________________________________________________
activation_1 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 8)                 4104
_________________________________________________________________
```



```
Train accuracy: 0.9800412
Train loss: 0.09062205221953351
Val accuracy: 0.96637523
Val loss: 0.12578026396137862
Test accuracy: 0.7309417
Test loss: 1.020298310994033
```

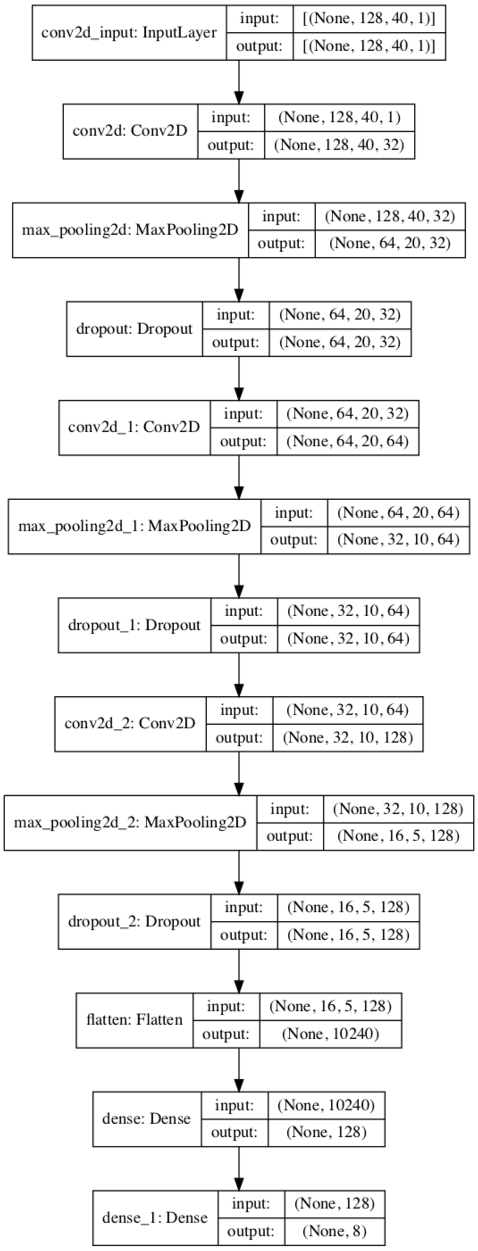
```
               precision    recall  f1-score   support

         COPD       1.00      0.83      0.91       199
      Healthy       0.49      0.55      0.52        69
         URTI       0.59      0.53      0.56        45
Bronchiectasis       0.91      0.94      0.92        31
 Bronchiolitis       0.32      0.80      0.45        25
    Pneumonia       0.73      0.66      0.69        73
         LRTI       0.00      0.00      0.00         3
       Asthma       1.00      1.00      1.00         1

  avg / total       0.78      0.73      0.75       446
```

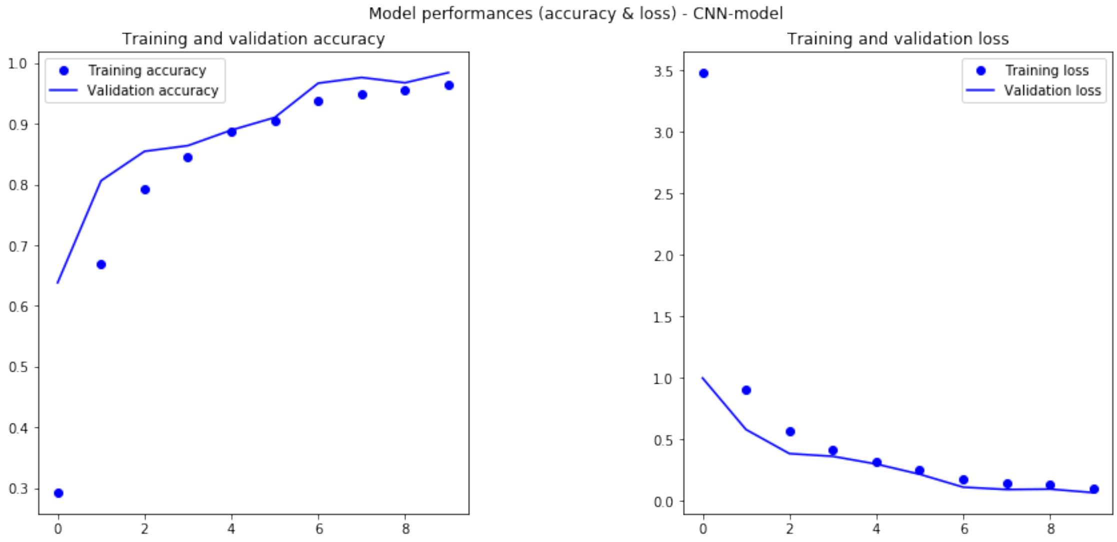


2.7.1.3.2.5 Convolutional Neural Networks (CNN)

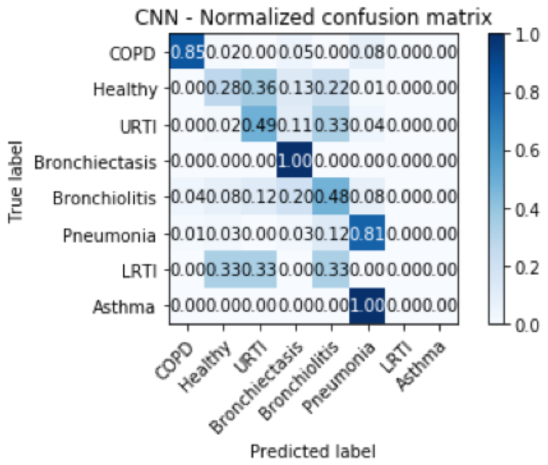
The results relative to CNN model are stacked in this chapter, for exhaustive information purpose.

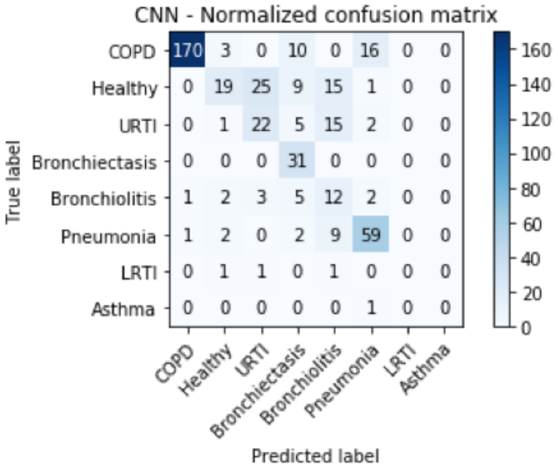
```
Model: "CNN-model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 128, 40, 32)       320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 64, 20, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 64, 20, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 64, 20, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 10, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 32, 10, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 10, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 5, 128)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 16, 5, 128)        0
_________________________________________________________________
flatten (Flatten)            (None, 10240)             0
_________________________________________________________________
dense (Dense)                (None, 128)               1310848
_________________________________________________________________
dense_1 (Dense)              (None, 8)                 1032
=================================================================
```


conv2d_input: InputLayer | input: [(None, 128, 40, 1)] | output: [(None, 128, 40, 1)]
conv2d: Conv2D | input: (None, 128, 40, 1) | output: (None, 128, 40, 32)
max_pooling2d: MaxPooling2D | input: (None, 128, 40, 32) | output: (None, 64, 20, 32)
dropout: Dropout | input: (None, 64, 20, 32) | output: (None, 64, 20, 32)
conv2d_1: Conv2D | input: (None, 64, 20, 32) | output: (None, 64, 20, 64)
max_pooling2d_1: MaxPooling2D | input: (None, 64, 20, 64) | output: (None, 32, 10, 64)
dropout_1: Dropout | input: (None, 32, 10, 64) | output: (None, 32, 10, 64)
conv2d_2: Conv2D | input: (None, 32, 10, 64) | output: (None, 32, 10, 128)
max_pooling2d_2: MaxPooling2D | input: (None, 32, 10, 128) | output: (None, 16, 5, 128)
dropout_2: Dropout | input: (None, 16, 5, 128) | output: (None, 16, 5, 128)
flatten: Flatten | input: (None, 16, 5, 128) | output: (None, 10240)
dense: Dense | input: (None, 10240) | output: (None, 128)
dense_1: Dense | input: (None, 128) | output: (None, 8)

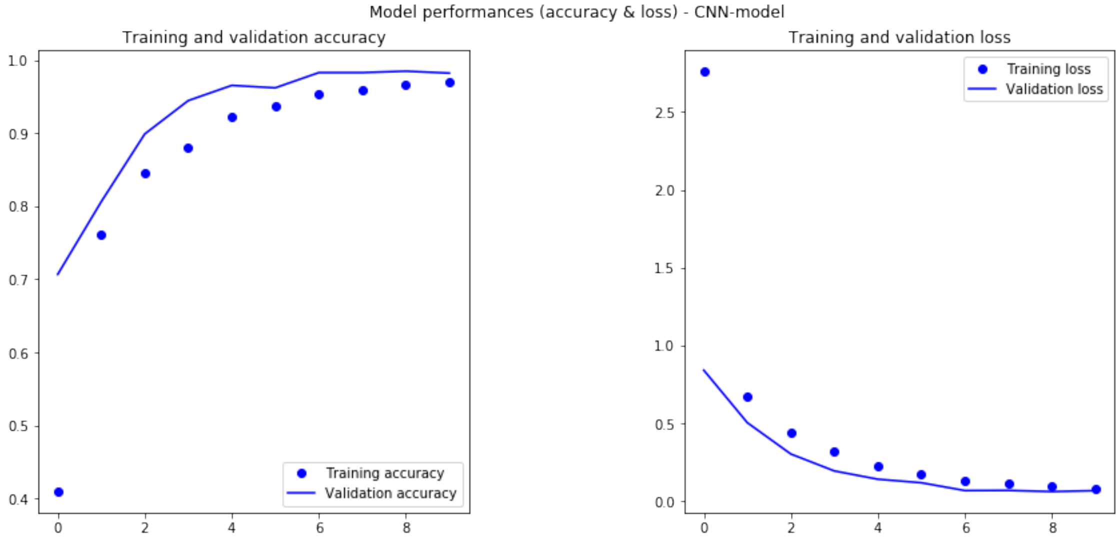

Model performances (accuracy & loss) - CNN-model
Training and validation accuracy
Training accuracy
Validation accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
2
4
6
8
Training and validation loss
Training loss
Validation loss
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
2
4
6
8

```
Val accuracy: 0.9845326
Val loss: 0.06750214426193361
Test accuracy: 0.80717486
Test loss: 0.6048165594515779

precision    recall  f1-score   support

          COPD       0.93      0.91      0.92       199
       Healthy       0.80      0.68      0.73        69
          URTI       0.56      0.67      0.61        45
Bronchiectasis       0.93      0.90      0.92        31
 Bronchiolitis       0.50      0.72      0.59        25
     Pneumonia       0.83      0.74      0.78        73
          LRTI       0.50      0.33      0.40         3
        Asthma       0.20      1.00      0.33         1

   avg / total       0.83      0.81      0.81       446
```





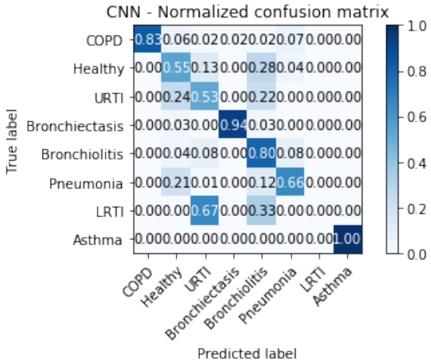
2eme

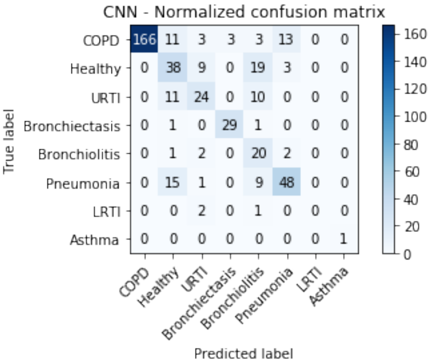


```
Train accuracy: 0.9954063
Train loss: 0.01837654190781805
```

```
Val accuracy: 0.98251516
Val loss: 0.06653076739980088
Test accuracy: 0.7735426
Test loss: 0.7832625452178477
```

```
              precision    recall  f1-score   support

        COPD       0.92      0.88      0.90       199
     Healthy       0.56      0.75      0.64        69
        URTI       0.52      0.51      0.52        45
Bronchiectasis     0.94      0.94      0.94        31
 Bronchiolitis     0.65      0.52      0.58        25
   Pneumonia       0.80      0.70      0.74        73
        LRTI       0.33      0.33      0.33         3
      Asthma       1.00      1.00      1.00         1

  avg / total      0.79      0.77      0.78       446
```

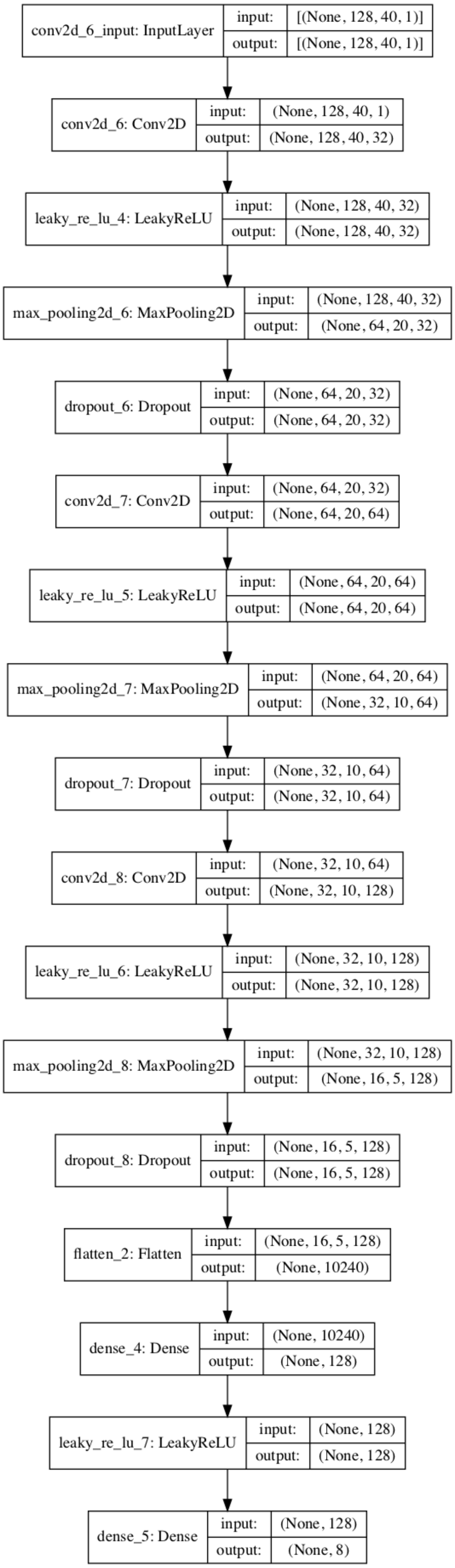


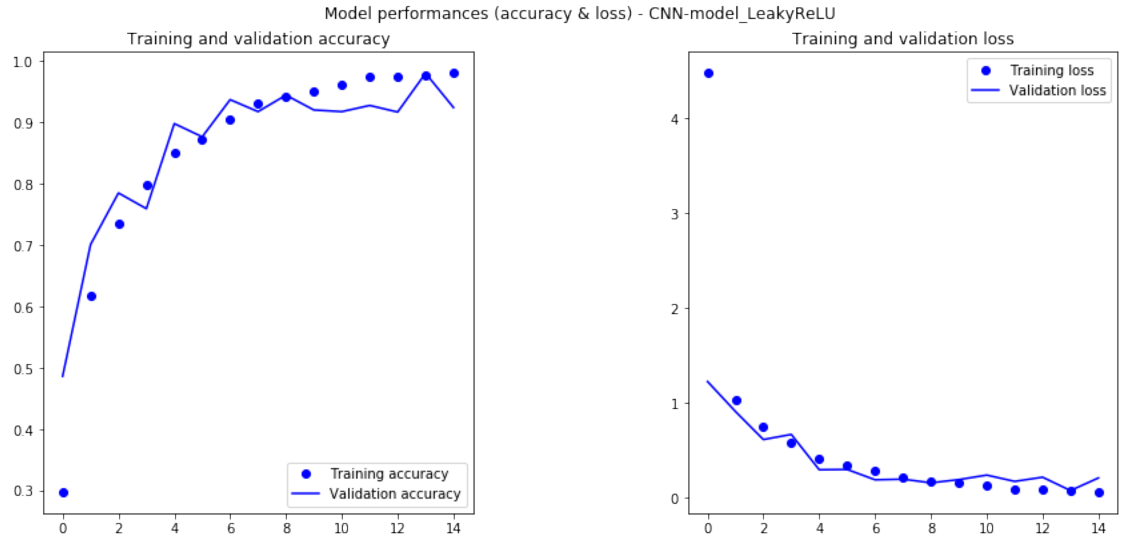


CNN LeakyReLU

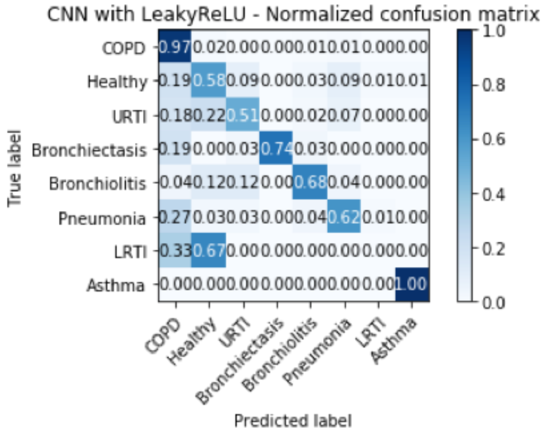
```
Model: "CNN-model_LeakyReLU"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_6 (Conv2D)            (None, 128, 40, 32)       320       
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 128, 40, 32)       0         
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 64, 20, 32)        0         
_________________________________________________________________
dropout_6 (Dropout)          (None, 64, 20, 32)        0         
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 64, 20, 64)        18496     
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 64, 20, 64)        0         
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 32, 10, 64)        0         
_________________________________________________________________
dropout_7 (Dropout)          (None, 32, 10, 64)        0         
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 32, 10, 128)       73856     
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 32, 10, 128)       0         
```

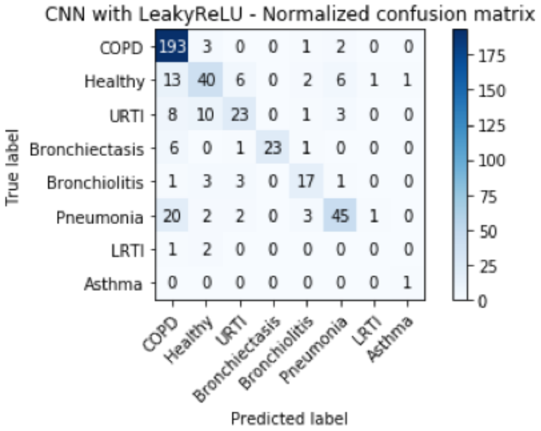
```
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 16, 5, 128)        0
_________________________________________________________________
dropout_8 (Dropout)          (None, 16, 5, 128)        0
_________________________________________________________________
flatten_2 (Flatten)          (None, 10240)             0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               1310848
_________________________________________________________________
leaky_re_lu_7 (LeakyReLU)    (None, 128)               0
_________________________________________________________________
dense_5 (Dense)              (None, 8)                 1032
=================================================================
```


conv2d_6_input: InputLayer
input: [(None, 128, 40, 1)]
output: [(None, 128, 40, 1)]
conv2d_6: Conv2D
input: (None, 128, 40, 1)
output: (None, 128, 40, 32)
leaky_re_lu_4: LeakyReLU
input: (None, 128, 40, 32)
output: (None, 128, 40, 32)
max_pooling2d_6: MaxPooling2D
input: (None, 128, 40, 32)
output: (None, 64, 20, 32)
dropout_6: Dropout
input: (None, 64, 20, 32)
output: (None, 64, 20, 32)
conv2d_7: Conv2D
input: (None, 64, 20, 32)
output: (None, 64, 20, 64)
leaky_re_lu_5: LeakyReLU
input: (None, 64, 20, 64)
output: (None, 64, 20, 64)
max_pooling2d_7: MaxPooling2D
input: (None, 64, 20, 64)
output: (None, 32, 10, 64)
dropout_7: Dropout
input: (None, 32, 10, 64)
output: (None, 32, 10, 64)
conv2d_8: Conv2D
input: (None, 32, 10, 64)
output: (None, 32, 10, 128)
leaky_re_lu_6: LeakyReLU
input: (None, 32, 10, 128)
output: (None, 32, 10, 128)
max_pooling2d_8: MaxPooling2D
input: (None, 32, 10, 128)
output: (None, 16, 5, 128)
dropout_8: Dropout
input: (None, 16, 5, 128)
output: (None, 16, 5, 128)
flatten_2: Flatten
input: (None, 16, 5, 128)
output: (None, 10240)
dense_4: Dense
input: (None, 10240)
output: (None, 128)
leaky_re_lu_7: LeakyReLU
input: (None, 128)
output: (None, 128)
dense_5: Dense
input: (None, 128)
output: (None, 8)

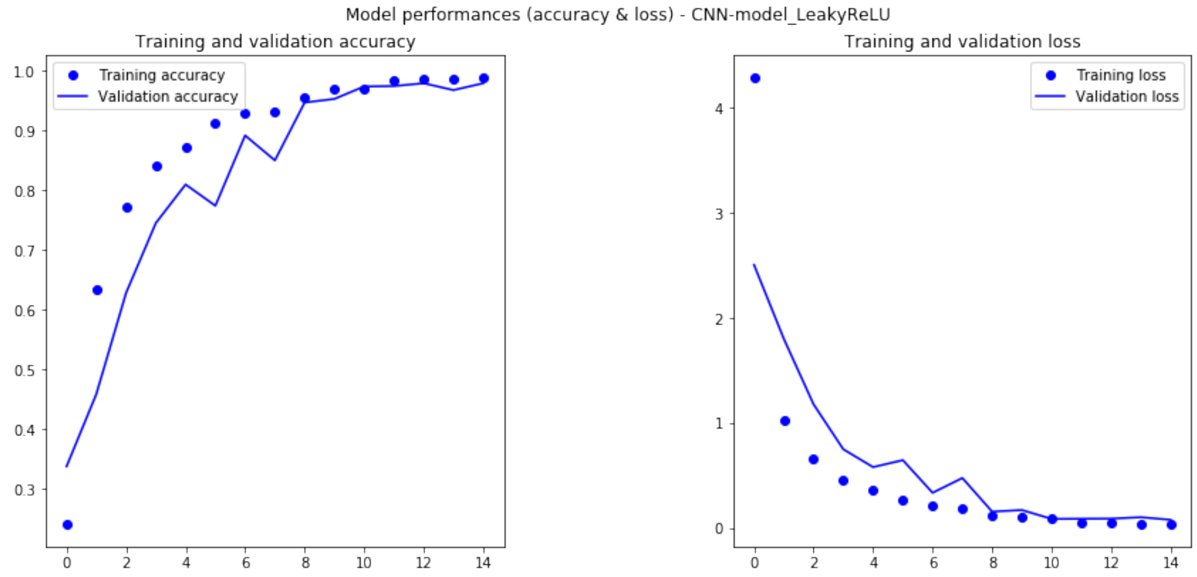


```
Val loss: 0.2061 - categorical_accuracy: 0.9240
Test loss: 0.9865 - categorical_accuracy: 0.7668
Test accuracy: 0.76681614
Test loss: 0.9864628127538035

precision    recall  f1-score   support

          COPD       0.80      0.97      0.88       199
       Healthy       0.67      0.58      0.62        69
          URTI       0.66      0.51      0.57        45
Bronchiectasis       1.00      0.74      0.85        31
 Bronchiolitis       0.68      0.68      0.68        25
     Pneumonia       0.79      0.62      0.69        73
          LRTI       0.00      0.00      0.00         3
        Asthma       0.50      1.00      0.67         1

   avg / total       0.76      0.77      0.76       446
```



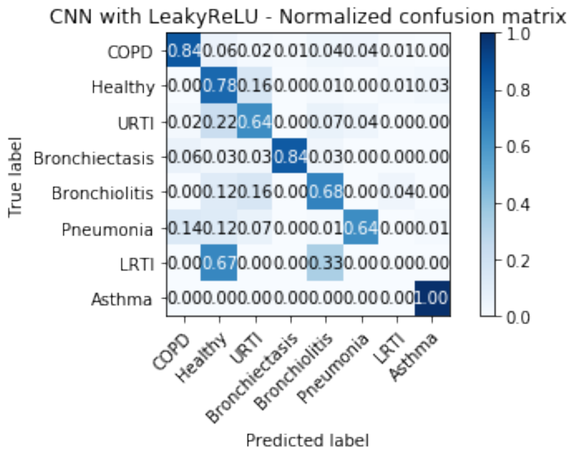


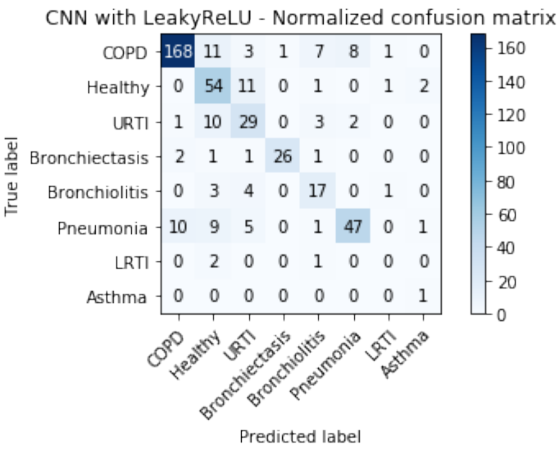
2nd run



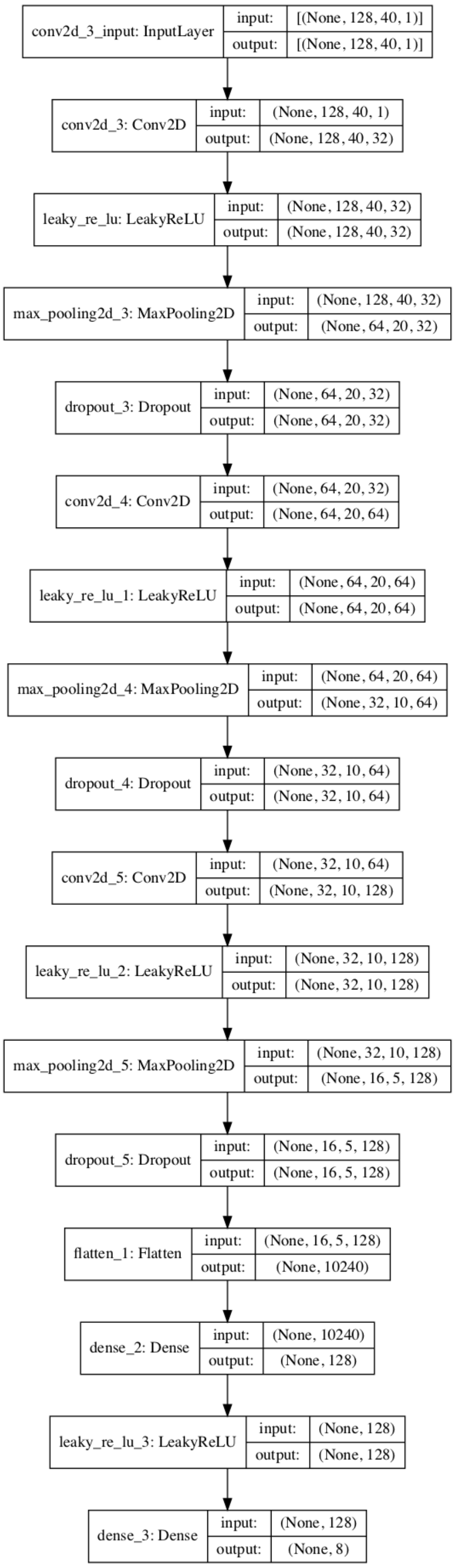
```
Train accuracy: 0.990971
Train loss: 0.034122208821784294
Val accuracy: 0.9791527
Val loss: 0.07943202468121549
Test accuracy: 0.76681614
Test loss: 0.9889184979579908
```

```
                precision    recall  f1-score   support

          COPD       0.93      0.84      0.88       199
       Healthy       0.60      0.78      0.68        69
          URTI       0.55      0.64      0.59        45
Bronchiectasis       0.96      0.84      0.90        31
 Bronchiolitis       0.55      0.68      0.61        25
     Pneumonia       0.82      0.64      0.72        73
          LRTI       0.00      0.00      0.00         3
        Asthma       0.25      1.00      0.40         1

   avg / total       0.80      0.77      0.77       446
```



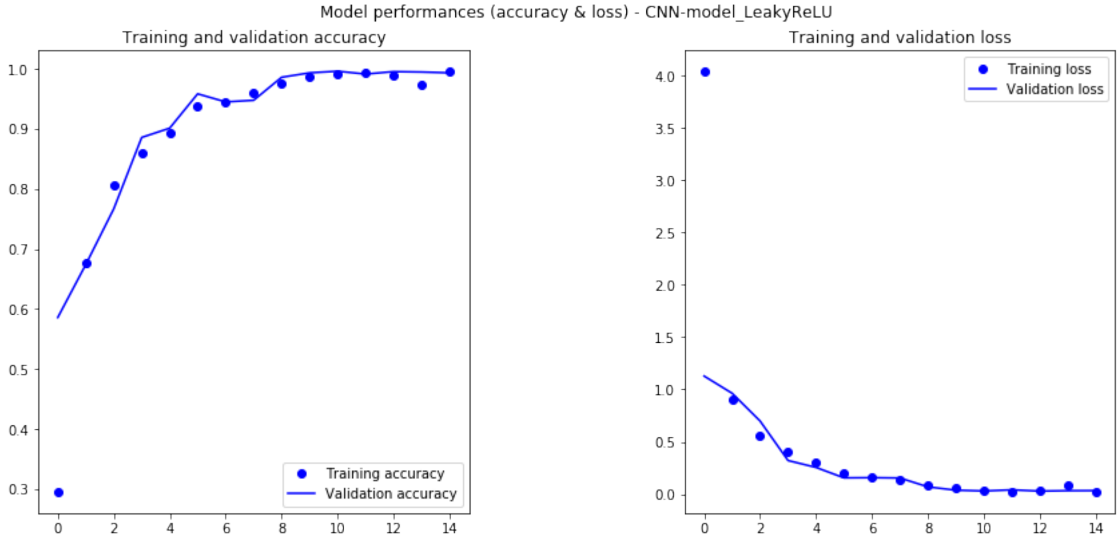


3rd run with Dropout layers coefficients reduced to 0.1.


conv2d_3_input: InputLayer
input: [(None, 128, 40, 1)]
output: [(None, 128, 40, 1)]
conv2d_3: Conv2D
input: (None, 128, 40, 1)
output: (None, 128, 40, 32)
leaky_re_lu: LeakyReLU
input: (None, 128, 40, 32)
output: (None, 128, 40, 32)
max_pooling2d_3: MaxPooling2D
input: (None, 128, 40, 32)
output: (None, 64, 20, 32)
dropout_3: Dropout
input: (None, 64, 20, 32)
output: (None, 64, 20, 32)
conv2d_4: Conv2D
input: (None, 64, 20, 32)
output: (None, 64, 20, 64)
leaky_re_lu_1: LeakyReLU
input: (None, 64, 20, 64)
output: (None, 64, 20, 64)
max_pooling2d_4: MaxPooling2D
input: (None, 64, 20, 64)
output: (None, 32, 10, 64)
dropout_4: Dropout
input: (None, 32, 10, 64)
output: (None, 32, 10, 64)
conv2d_5: Conv2D
input: (None, 32, 10, 64)
output: (None, 32, 10, 128)
leaky_re_lu_2: LeakyReLU
input: (None, 32, 10, 128)
output: (None, 32, 10, 128)
max_pooling2d_5: MaxPooling2D
input: (None, 32, 10, 128)
output: (None, 16, 5, 128)
dropout_5: Dropout
input: (None, 16, 5, 128)
output: (None, 16, 5, 128)
flatten_1: Flatten
input: (None, 16, 5, 128)
output: (None, 10240)
dense_2: Dense
input: (None, 10240)
output: (None, 128)
leaky_re_lu_3: LeakyReLU
input: (None, 128)
output: (None, 128)
dense_3: Dense
input: (None, 128)
output: (None, 8)

```
Model: "CNN-model_LeakyReLU"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 128, 40, 32)       320
```

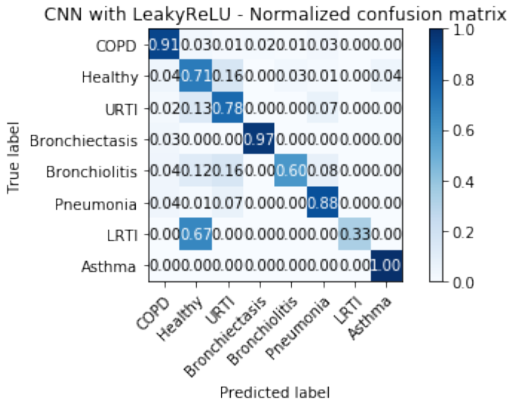
```
_________________________________________________________________
leaky_re_lu (LeakyReLU)      (None, 128, 40, 32)       0         
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 64, 20, 32)        0         
_________________________________________________________________
dropout_3 (Dropout)          (None, 64, 20, 32)        0         
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 64, 20, 64)        18496     
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 64, 20, 64)        0         
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 32, 10, 64)        0         
_________________________________________________________________
dropout_4 (Dropout)          (None, 32, 10, 64)        0         
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 32, 10, 128)       73856     
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 32, 10, 128)       0         
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 16, 5, 128)        0         
_________________________________________________________________
dropout_5 (Dropout)          (None, 16, 5, 128)        0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 10240)             0         
_________________________________________________________________
dense_2 (Dense)              (None, 128)               1310848   
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 128)               0         
_________________________________________________________________
dense_3 (Dense)              (None, 8)                 1032      
=================================================================
```

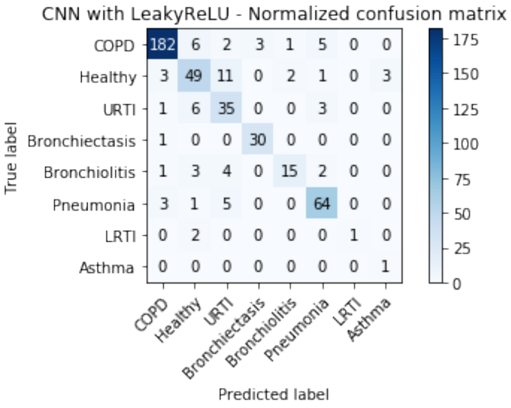


```
              precision    recall  f1-score   support

        COPD       0.95      0.91      0.93       199
     Healthy       0.73      0.71      0.72        69
```

```
          URTI       0.61      0.78      0.69        45
Bronchiectasis       0.91      0.97      0.94        31
 Bronchiolitis       0.83      0.60      0.70        25
     Pneumonia       0.85      0.88      0.86        73
          LRTI       1.00      0.33      0.50         3
        Asthma       0.25      1.00      0.40         1

   avg / total       0.86      0.85      0.85       446
```

```
Train accuracy: 0.9996832
Train loss: 0.0035000012984810445
Val accuracy: 0.99327505
Val loss: 0.03351340858136307
Test accuracy: 0.8452915
Test loss: 0.6733562957010997
```

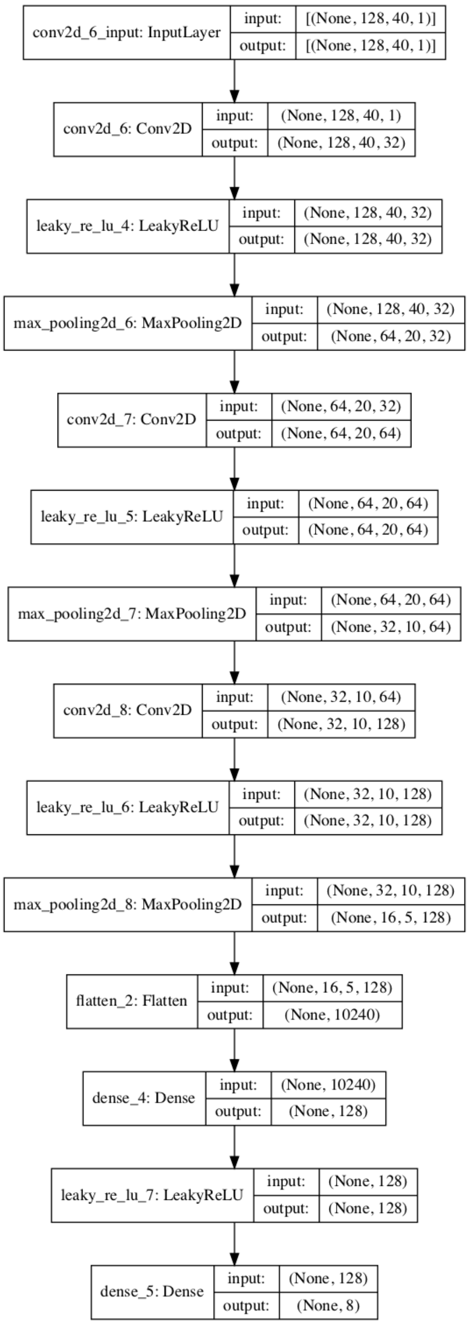


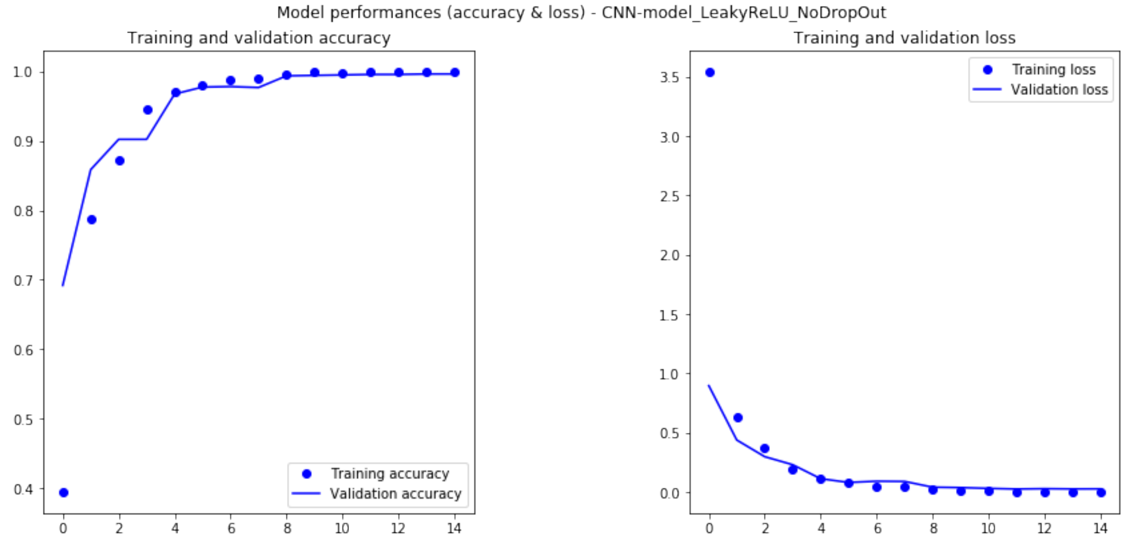


CNN Model LeakyReLU no Drop

```
Model: "CNN-model_LeakyReLU_NoDropOut"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_6 (Conv2D)            (None, 128, 40, 32)       320
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 128, 40, 32)       0
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 64, 20, 32)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 64, 20, 64)        18496
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 64, 20, 64)        0
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 32, 10, 64)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 32, 10, 128)       73856
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 32, 10, 128)       0
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 16, 5, 128)        0
_________________________________________________________________
```

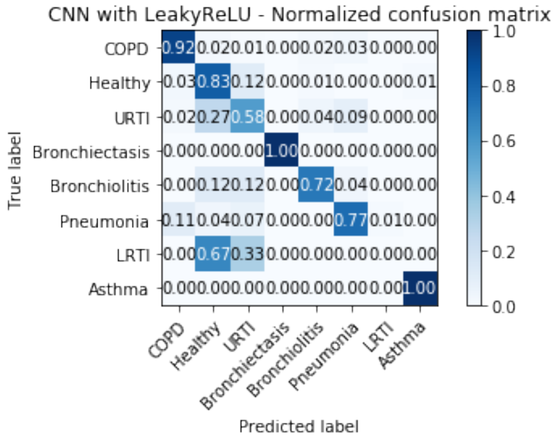
```
flatten_2 (Flatten)          (None, 10240)            0
_________________________________________________________________
dense_4 (Dense)              (None, 128)              1310848
_________________________________________________________________
leaky_re_lu_7 (LeakyReLU)    (None, 128)              0
_________________________________________________________________
dense_5 (Dense)              (None, 8)                1032
=================================================================
```


conv2d_6_input: InputLayer
input: [(None, 128, 40, 1)]
output: [(None, 128, 40, 1)]
conv2d_6: Conv2D
input: (None, 128, 40, 1)
output: (None, 128, 40, 32)
leaky_re_lu_4: LeakyReLU
input: (None, 128, 40, 32)
output: (None, 128, 40, 32)
max_pooling2d_6: MaxPooling2D
input: (None, 128, 40, 32)
output: (None, 64, 20, 32)
conv2d_7: Conv2D
input: (None, 64, 20, 32)
output: (None, 64, 20, 64)
leaky_re_lu_5: LeakyReLU
input: (None, 64, 20, 64)
output: (None, 64, 20, 64)
max_pooling2d_7: MaxPooling2D
input: (None, 64, 20, 64)
output: (None, 32, 10, 64)
conv2d_8: Conv2D
input: (None, 32, 10, 64)
output: (None, 32, 10, 128)
leaky_re_lu_6: LeakyReLU
input: (None, 32, 10, 128)
output: (None, 32, 10, 128)
max_pooling2d_8: MaxPooling2D
input: (None, 32, 10, 128)
output: (None, 16, 5, 128)
flatten_2: Flatten
input: (None, 16, 5, 128)
output: (None, 10240)
dense_4: Dense
input: (None, 10240)
output: (None, 128)
leaky_re_lu_7: LeakyReLU
input: (None, 128)
output: (None, 128)
dense_5: Dense
input: (None, 128)
output: (None, 8)

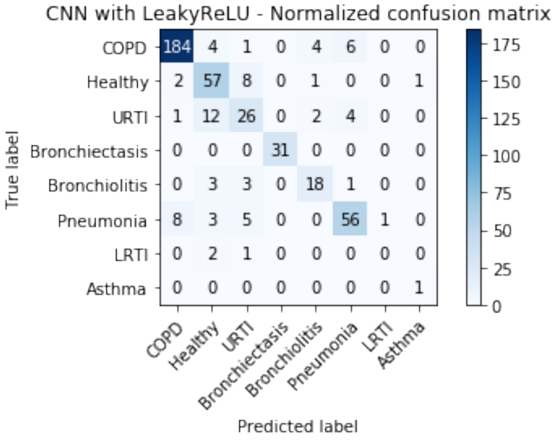


```
Train accuracy: 1.0
Train loss: 0.0006165395029799963
Val accuracy: 0.9966375
Val loss: 0.028135972805891942
Test accuracy: 0.83632284
Test loss: 0.7555456533025733
```

```
                precision    recall  f1-score   support

          COPD       0.94      0.92      0.93       199
       Healthy       0.70      0.83      0.76        69
          URTI       0.59      0.58      0.58        45
Bronchiectasis       1.00      1.00      1.00        31
 Bronchiolitis       0.72      0.72      0.72        25
     Pneumonia       0.84      0.77      0.80        73
          LRTI       0.00      0.00      0.00         3
        Asthma       0.50      1.00      0.67         1

   avg / total       0.84      0.84      0.84       446
```



	COPD	Healthy	URTI	Bronchiectasis	Bronchiolitis	Pneumonia	LRTI	Asthma
COPD	0.92	0.02	0.01	0.00	0.02	0.03	0.00	0.00
Healthy	0.03	0.83	0.12	0.00	0.01	0.00	0.00	0.01
URTI	0.02	0.27	0.58	0.00	0.04	0.09	0.00	0.00
Bronchiectasis	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
Bronchiolitis	0.00	0.12	0.12	0.00	0.72	0.04	0.00	0.00
Pneumonia	0.11	0.04	0.07	0.00	0.00	0.77	0.01	0.00
LRTI	0.00	0.67	0.33	0.00	0.00	0.00	0.00	0.00
Asthma	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00



	COPD	Healthy	URTI	Bronchiectasis	Bronchiolitis	Pneumonia	LRTI	Asthma
COPD	184	4	1	0	4	6	0	0
Healthy	2	57	8	0	1	0	0	1
URTI	1	12	26	0	2	4	0	0
Bronchiectasis	0	0	0	31	0	0	0	0
Bronchiolitis	0	3	3	0	18	1	0	0
Pneumonia	8	3	5	0	0	56	1	0
LRTI	0	2	1	0	0	0	0	0
Asthma	0	0	0	0	0	0	0	1

2.7.1.4 Synthesis

The table below synthetize the overall best performance obtained from the models of interest.

Model	Parameters	Train Accuracy	Train Loss	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss	Precision	Recall	F1-score
CNN-model_LeakyReLU_NoDropOut		1,000	0,001	0,997	0,028	0,836	0,756	0,84	0,84	0,84
CNN-model_LeakyReLU	**Dropout=0.1**	**1,000**	**0,004**	**0,993**	**0,034**	**0,845**	**0,673**	**0,86**	**0,85**	**0,85**
CNN-model_LeakyReLU	Dropout=0.3	0,991	0,034	0,979	0,079	0,767	0,989	0,79	0,77	0,78
CNN-model	Dropout=0.3			0,985	0,068	0,807	0,605	0,83	0,81	0,81
DFF-NN	Dropout=0.5	0,980	0,091	0,966	0,126	0,731	1,020	0,78	0,73	0,75
SVM	poly, degree=8					0,740		0,74	0,74	0,74

The best prediction model experimented are so far the CNN models with LeakyReLU and a weak Dropout layer with dropout coefficient 0.1, in term of prediction accuracy, balance across diagnosis types, and stability. Model with LeakyReLU and no dropout is almost at the same level, with a slight difference. More iterations (Monte Carlo for example) would be needed to evaluate one against the other.

The model performances are remained below.

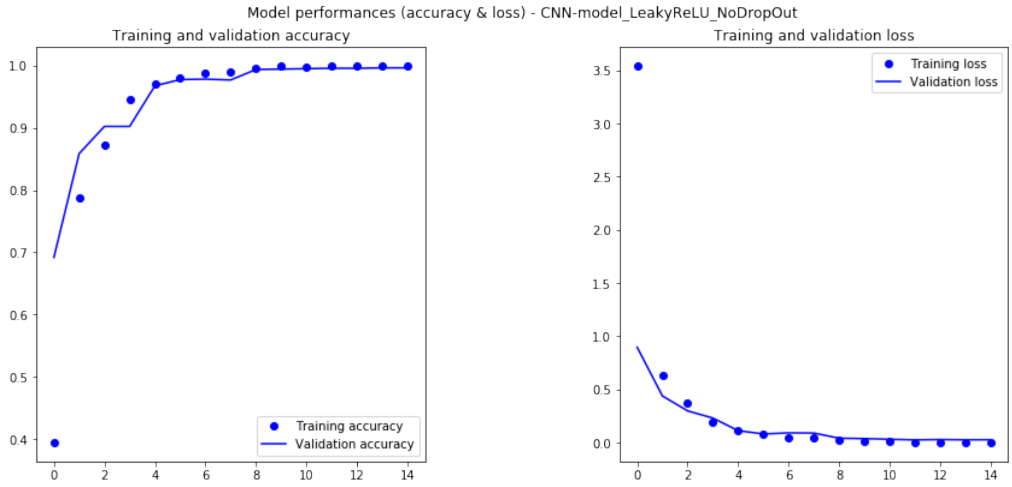
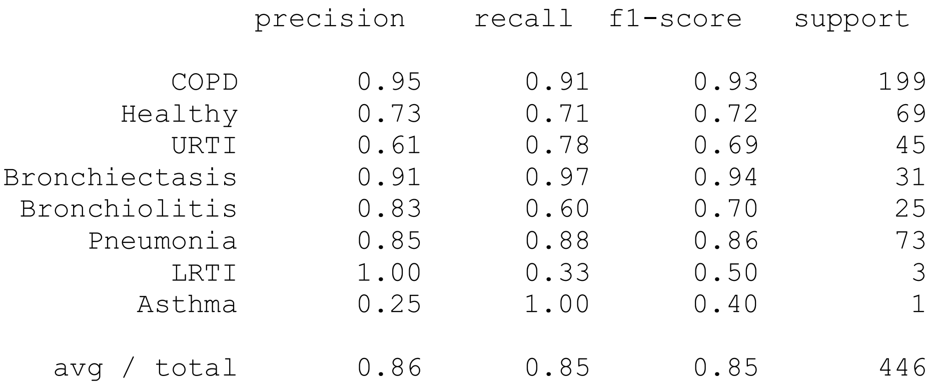


Figure 8- LeakyReLU with dropout=0.1 Train & Validation accuracy/loss

Both train and validation curves show a good fit.



```
               precision    recall  f1-score   support

         COPD       0.95      0.91      0.93       199
      Healthy       0.73      0.71      0.72        69
         URTI       0.61      0.78      0.69        45
Bronchiectasis       0.91      0.97      0.94        31
 Bronchiolitis       0.83      0.60      0.70        25
    Pneumonia       0.85      0.88      0.86        73
         LRTI       1.00      0.33      0.50         3
       Asthma       0.25      1.00      0.40         1

  avg / total       0.86      0.85      0.85       446
```

Figure 9 – LeakyReLU with dropout=0.1 : classification report

The model has overall good performances excellent performances in both precision and recall on COPD, Bronchiectasis and Pneumonia, and overall good performances on other categories, with a relative balance in confusion between false positives and true negatives.

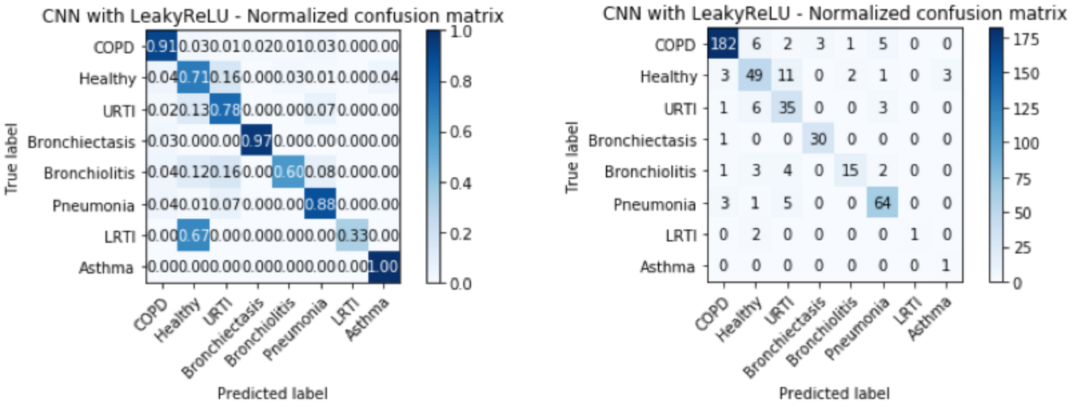


Figure 10 - LeakyReLU with dropout=0.1 Confusion Matrix

The model performances have been improved mainly by:

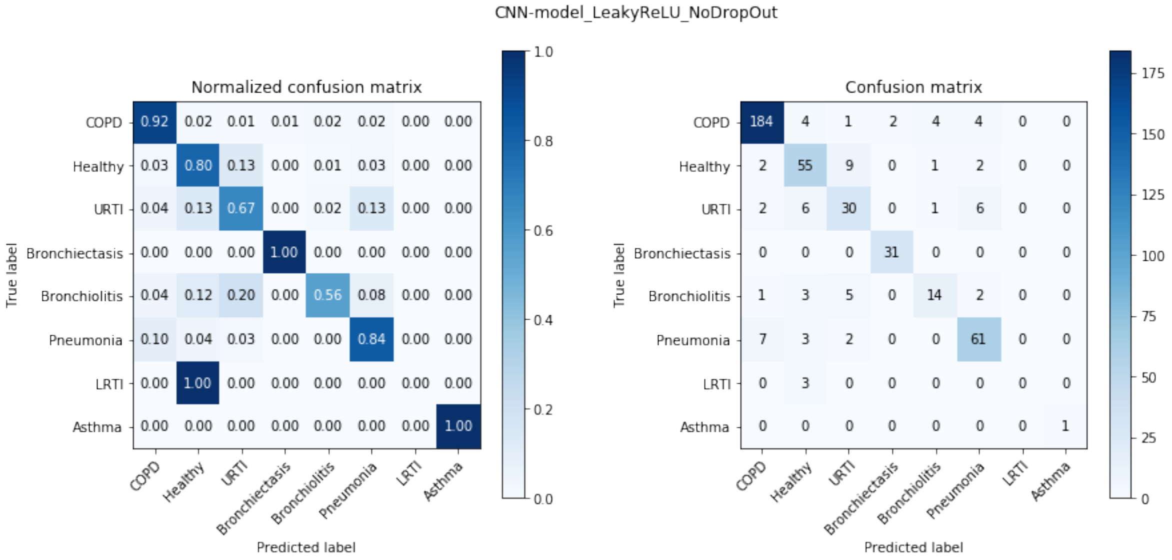
- balancing the data by data augmentation techniques,
- for SVM, adapting the kernel strategy (poly, degree =8)
- for DFF-NN and CNNs, by adapting parameters such as batch size, drop coefficient of the Dropout layers, number of epochs, and adapting activation functions (such as ReLU vs LeakyReLU).

CNN models converge much faster than Feed forward neural network.

The CNN models with LeakyReLU and no dropout provide the best prediction performances while being more robust to the variations through the different runs.

```
                precision    recall  f1-score   support

          COPD       0.94      0.92      0.93       199
       Healthy       0.74      0.80      0.77        69
          URTI       0.64      0.67      0.65        45
Bronchiectasis       0.94      1.00      0.97        31
 Bronchiolitis       0.70      0.56      0.62        25
     Pneumonia       0.81      0.84      0.82        73
          LRTI       0.00      0.00      0.00         3
        Asthma       1.00      1.00      1.00         1

   avg / total       0.84      0.84      0.84       446
```

Figure 11- Prediction performance - CNN with LeakyReLU and no dropout



Normalized confusion matrix

	COPD	Healthy	URTI	Bronchiectasis	Bronchiolitis	Pneumonia	LRTI	Asthma
COPD	0.92	0.02	0.01	0.01	0.02	0.02	0.00	0.00
Healthy	0.03	0.80	0.13	0.00	0.01	0.03	0.00	0.00
URTI	0.04	0.13	0.67	0.00	0.02	0.13	0.00	0.00
Bronchiectasis	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
Bronchiolitis	0.04	0.12	0.20	0.00	0.56	0.08	0.00	0.00
Pneumonia	0.10	0.04	0.03	0.00	0.00	0.84	0.00	0.00
LRTI	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
Asthma	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00

Confusion matrix

	COPD	Healthy	URTI	Bronchiectasis	Bronchiolitis	Pneumonia	LRTI	Asthma
COPD	184	4	1	2	4	4	0	0
Healthy	2	55	9	0	1	2	0	0
URTI	2	6	30	0	1	6	0	0
Bronchiectasis	0	0	0	31	0	0	0	0
Bronchiolitis	1	3	5	0	14	2	0	0
Pneumonia	7	3	2	0	0	61	0	0
LRTI	0	3	0	0	0	0	0	0
Asthma	0	0	0	0	0	0	0	1

Figure 12 - Another run of CNN with LeakyReLU and no dropout

The best boost in performance was given by data augmentation (in general), the kernel adjustment (for SVM), and a combination of the use of the LeakyReLU activation function and drop coefficients adjustment (for neural networks).

2.7.2 Justification

Random Forest and Deep Feed Forward neural network are chosen to be reference “weak” learners for the ML and DL approach respectively.
SVM choice is justified by its characteristics of efficiency with a small datasets and ability to deal with noisy data and outliers.

The scikit-learn (0.19.1) SVM implementation was chosen because of its native support of multi-class and non-linear kernels, compared to the Spark MLLib implementation.

Convolutional Neural Networks were chosen in line with the MFCC feature transformation, with give an image-like vision of audio signals, and the performances of CNN regarding image classification .

Keras/Tensorflow choice is based on knowledge of the technologies.
Versions used are:

- Keras 2.2.2
- Tensorflow 2.0.0a0

The LeakyReLU activation function choice was driven by the variability in prediction performances of the CNN model with Dropout and ReLU. ReLU seemed to make cells dying too fast, thus implying the model to drive choices based on the content of the training dataset (overfitting). The LeakyReLU activation function is an attempt to solve dying ReLU.

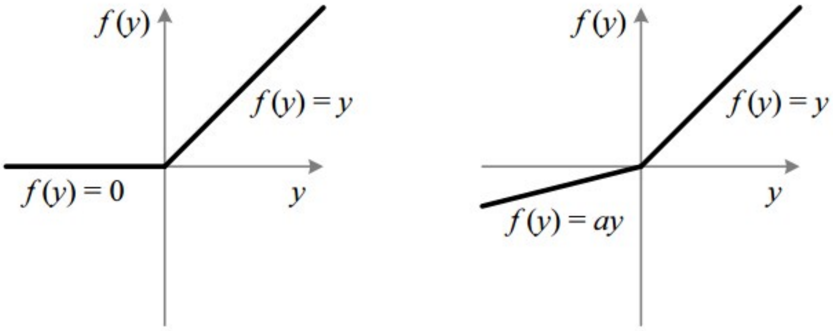


Figure 13- ReLU vs. LeakyReLU (source: SAGAR SHARM, https://towardsdatascience.com/)

2.8 Applications / Data Products

2.8.1 Technology Choice

The data products defined are the set of Jupyter notebooks, available on GitHub:
https://github.com/bnassivet/IBM_Advanced_DataScience_Capstone

Execution of the notebooks also provide:

- the exported Keras/Tensorflow trained models,
- Feature dataset for further model evaluation.

2.8.2 Justification

The cycle is the first cycle on the Respiratory Disorder dataset. The goal of this first cycle was an experimentation for Proof of Concept on the capability to classify the diagnosis based on Supervised Learning method, using either ML or DL approaches. Jupyter notebooks are an efficient way to test, document and share the different steps of the processus, along with foundation of the still-to-be-improved pipelines.

2.9 Security, Information Governance and Systems Management

2.9.1 Technology Choice

This component is not needed in the first step.

2.9.2 Justification

In the first step, the dataset is static with data anonymized. Thus no Information security, governance or systems management is required.
These architecture building blocks will become critical at the time the system deployment will be addressed, due to the private nature of the data and processing outputs.