

Universidad
Industrial de
Santander



Escuela de
Ingeniería de
Sistemas e
Informática



Detección de Neumonía Empleando Radiografías de Tórax a través de redes Neuronales

Proyecto inteligencia artificial

Gómez Natalia¹, Ramírez Nicolas¹

¹Dept. of Computer Science
Universidad Industrial de Santander

April 1, 2020

1 Introduction

- Motivación
- Data-set

2 Metodo propuesto

- Metodología
- Clasificadores

3 Resultados

- Resultados Preliminares
- Resultados finales

4 Conclusiones

1 Introduction

- Motivación
- Data-set

2 Metodo propuesto

- Metodología
- Clasificadores

3 Resultados

- Resultados Preliminares
- Resultados finales

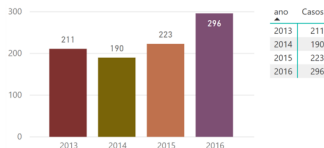
4 Conclusiones

¿NEUMONIA?

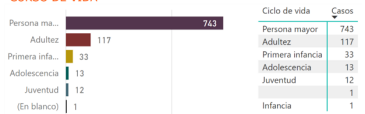
- Esta enfermedad es una de las principales causas de mortalidad tanto en niños como en adultos en todo el mundo.
- Ambigüedad entre profesionales.

Causa carga enfermedad	Casos
Cardiopatía isquémica	2.530
Enfermedad cerebrovascular	1.397
Infecciones de vías respiratorias inferiores	920
Enf. Pulmonar obstructiva crónica	795
Diabetes mellitus	767
Otras enfermedades del sistema circulatorio **	564
Residuo	537
Cáncer de Estomago	493
Cardiopatía hipertensiva	457
Agresiones	450
Otras enfermedades respiratorias **	423
Nefritis y nefrosis	383
Otras enfermedades del sistema digestivo **	378
Accidentes de tránsito	369
Cáncer de traquea, bronquios,pulmón	363

MUERTES POR AÑO



CURSO DE VIDA



1 Introduction

- Motivación
- Data-set

2 Metodo propuesto

- Metodología
- Clasificadores

3 Resultados

- Resultados Preliminares
- Resultados finales

4 Conclusiones

Introduction

CHEST X-RAY IMAGES (PNEUMONIA)

- Dataset extraído de <https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia>

	Normal	Neumonia	Total
Train	1,341	3,875	5,216
Test	234	390	624

Table: Cantidad de imagenes utilizadas para entrenar y testear



Procesamiento

- Listar
- Recorrer las imágenes.
- Convertir a escalas de grises.
- Dimensionar las imágenes.
- Guardar.
- Etiquetar

1 Introduction

- Motivación
- Data-set

2 Metodo propuesto

- Metodología
- Clasificadores

3 Resultados

- Resultados Preliminares
- Resultados finales

4 Conclusiones

Model: "sequential_13"

Layer (type)	Output Shape	Param #
flatten_13 (Flatten)	(None, 22500)	0
dense_69 (Dense)	(None, 200)	4500200
dense_70 (Dense)	(None, 100)	20100
dense_71 (Dense)	(None, 150)	15150
dense_72 (Dense)	(None, 50)	7550
dense_73 (Dense)	(None, 2)	102
Total params: 4,543,102		
Trainable params: 4,543,102		
Non-trainable params: 0		

1 Introduction

- Motivación
- Data-set

2 Metodo propuesto

- Metodología
- Clasificadores

3 Resultados

- Resultados Preliminares
- Resultados finales

4 Conclusiones

Clasificadores

- Decision Tree Classifier
- Gaussian NB
- Random Forest
- SVM

1 Introduction

- Motivación
- Data-set

2 Metodo propuesto

- Metodología
- Clasificadores

3 Resultados

- Resultados Preliminares
- Resultados finales

4 Conclusiones

Resultados

```
[175] history=model.fit(data_test, ytest, epochs=10, batch_size=50)
```

Train on 624 samples

```
Epoch 1/10  
624/624 [=====] - 0s 666us/sample - loss: 51.4111 - acc: 0.7436  
Epoch 2/10  
624/624 [=====] - 0s 430us/sample - loss: 14.7225 - acc: 0.8045  
Epoch 3/10  
624/624 [=====] - 0s 450us/sample - loss: 5.5259 - acc: 0.8814  
Epoch 4/10  
624/624 [=====] - 0s 442us/sample - loss: 1.8953 - acc: 0.9391  
Epoch 5/10  
624/624 [=====] - 0s 444us/sample - loss: 2.1632 - acc: 0.9167  
Epoch 6/10  
624/624 [=====] - 0s 444us/sample - loss: 1.7931 - acc: 0.9327  
Epoch 7/10  
624/624 [=====] - 0s 434us/sample - loss: 6.6519 - acc: 0.8253  
Epoch 8/10  
624/624 [=====] - 0s 435us/sample - loss: 2.3851 - acc: 0.9103  
Epoch 9/10  
624/624 [=====] - 0s 432us/sample - loss: 2.3200 - acc: 0.9119  
Epoch 10/10  
624/624 [=====] - 0s 420us/sample - loss: 1.0129 - acc: 0.9471
```

```
test_loss, test_acc = model.evaluate(data_test, ytest)
```

```
624/624 [=====] - 0s 331us/sample - loss: 0.4802 - acc: 0.9696
```

Resultados

```
[179] history=model.fit(data_test, ytest, epochs=10 batch_size=200)
```

Train on 624 samples

```
Epoch 1/10  
624/624 [=====] - 0s 281us/sample - loss: 1.1599 - acc: 0.9327  
Epoch 2/10  
624/624 [=====] - 0s 275us/sample - loss: 1.0026 - acc: 0.9263  
Epoch 3/10  
624/624 [=====] - 0s 290us/sample - loss: 0.6228 - acc: 0.9503  
Epoch 4/10  
624/624 [=====] - 0s 279us/sample - loss: 0.1136 - acc: 0.9904  
Epoch 5/10  
624/624 [=====] - 0s 271us/sample - loss: 0.3721 - acc: 0.9599  
Epoch 6/10  
624/624 [=====] - 0s 279us/sample - loss: 0.1273 - acc: 0.9824  
Epoch 7/10  
624/624 [=====] - 0s 273us/sample - loss: 0.1289 - acc: 0.9776  
Epoch 8/10  
624/624 [=====] - 0s 289us/sample - loss: 0.2221 - acc: 0.9663  
Epoch 9/10  
624/624 [=====] - 0s 304us/sample - loss: 0.2336 - acc: 0.9696  
Epoch 10/10  
624/624 [=====] - 0s 284us/sample - loss: 0.5300 - acc: 0.9471
```

```
test_loss, test_acc = model.evaluate(data_test, ytest)
```

```
624/624 [=====] - 0s 215us/sample - loss: 3.7913 - acc: 0.7997
```

```
[185] history=model.fit(data_test, ytest, epochs=10)
```

Train on 624 samples

```
Epoch 1/10
624/624 [-----] - 0s 581us/sample - loss: 0.6587 - acc: 0.6250
Epoch 2/10
624/624 [-----] - 0s 578us/sample - loss: 0.6587 - acc: 0.6250
Epoch 3/10
624/624 [-----] - 0s 584us/sample - loss: 0.6587 - acc: 0.6250
Epoch 4/10
624/624 [-----] - 0s 586us/sample - loss: 0.6586 - acc: 0.6250
Epoch 5/10
624/624 [-----] - 0s 572us/sample - loss: 0.6586 - acc: 0.6250
Epoch 6/10
624/624 [-----] - 0s 580us/sample - loss: 0.6587 - acc: 0.6250
Epoch 7/10
624/624 [-----] - 0s 585us/sample - loss: 0.6587 - acc: 0.6250
Epoch 8/10
624/624 [-----] - 0s 559us/sample - loss: 0.6586 - acc: 0.6250
Epoch 9/10
624/624 [-----] - 0s 576us/sample - loss: 0.6588 - acc: 0.6250
Epoch 10/10
624/624 [-----] - 0s 566us/sample - loss: 0.6586 - acc: 0.6250
```

```
test_loss, test_acc = model.evaluate(data_test, ytest)
```

```
624/624 [-----] - 0s 217us/sample - loss: 0.6585 - acc: 0.6250
```

```
[183] history=model.fit(data_test, ytest, epochs=10, batch_size=5)
```

Train on 624 samples

```
Epoch 1/10
624/624 [-----] - 2s 3ms/sample - loss: 5.9328 - acc: 0.8141
Epoch 2/10
624/624 [-----] - 2s 3ms/sample - loss: 0.7868 - acc: 0.5705
Epoch 3/10
624/624 [-----] - 2s 3ms/sample - loss: 1.8227 - acc: 0.6250
Epoch 4/10
624/624 [-----] - 2s 3ms/sample - loss: 0.6617 - acc: 0.6250
Epoch 5/10
624/624 [-----] - 2s 3ms/sample - loss: 0.6603 - acc: 0.6250
Epoch 6/10
624/624 [-----] - 2s 3ms/sample - loss: 0.6590 - acc: 0.6250
Epoch 7/10
624/624 [-----] - 2s 3ms/sample - loss: 0.6591 - acc: 0.6250
Epoch 8/10
624/624 [-----] - 2s 3ms/sample - loss: 0.6591 - acc: 0.6250
Epoch 9/10
624/624 [-----] - 2s 3ms/sample - loss: 0.6592 - acc: 0.6250
Epoch 10/10
624/624 [-----] - 2s 3ms/sample - loss: 0.6595 - acc: 0.6250
```

```
test_loss, test_acc = model.evaluate(data_test, ytest)
```

```
624/624 [-----] - 0s 235us/sample - loss: 0.6585 - acc: 0.6250
```

Resultados

CASIFICADOR	PRECISIÓN
DecisionTreeClassifier	82.9 %
GaussianNB	85.6%
DNN	96.9%

Schedule

1 Introduction

- Motivación
- Data-set

2 Metodo propuesto

- Metodología
- Clasificadores

3 Resultados

- Resultados Preliminares
- Resultados finales

4 Conclusiones

Resultados

CASIFICADOR	PRECISIÓN
DecisionTreeClassifier	85.9 %
GaussianNB	85.6%
Random Forest Classifier	94.3%
SVM	85.6%
DNN	89.7%

Conclusiones

- Esperabamos un rendimiento sobresaliente con la red neuronal por ser en estos momentos tendencia mundial, pero los resultados obtenidos muestran una mejor precisión con otros metodos de vanguardia.
- La red neuronal para este caso no fue la más precisa, pero en cuestión de tiempo-precisión es la mejor opción, se debe tener en cuenta, ya que toma relevancia a la hora de trabajar con muchos datos.
- Se determinó que el clasificador que arrojó mejores resultados fue Random Forest Classifier