- Reconocimiento sonoro de instrumentos musicales

Universidad Internacional de Valencia

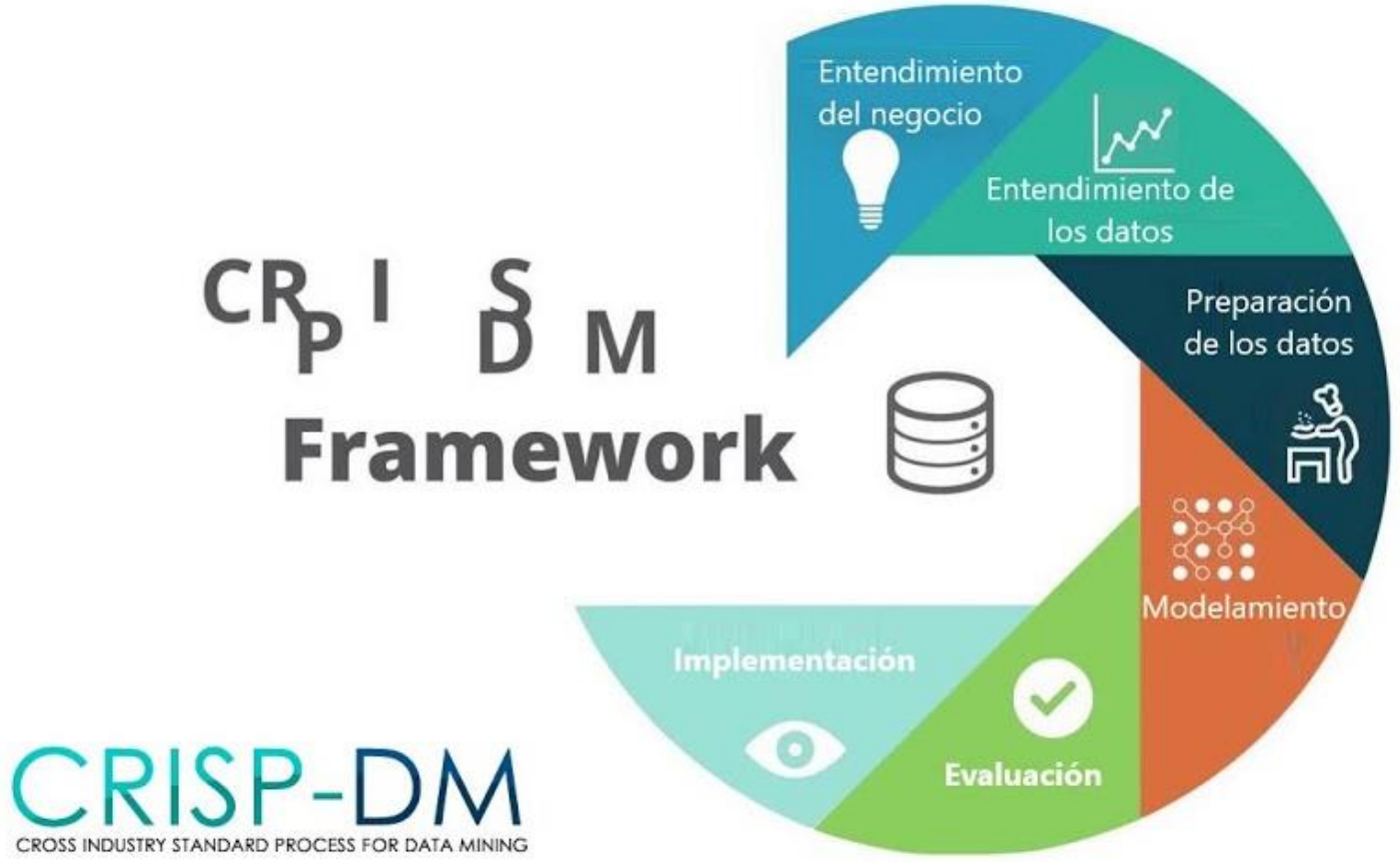
Autor: William Steve Rodriguez Villamizar

De:
Planeta Formación y Universidades

Objetivos

- Seleccionar un dataset adecuado para el problema.
- Aplicar una serie de transformaciones sobre los audios para adecuarlos para el modelo de IA.
- Entrenar un modelo de IA para clasificación de instrumentos musicales en un audio de 2 segundos.
- Implantar el modelo en producción casi en RT

Metodología a aplicar



Entendimiento de los datos

Subtítulo presentación

Dataset

- John Thickstun, Zaid Harchaoui y Sham Kakade tienen su dataset de musinet en kaggle

330 Archivos

+1M etiquetas
Instrumentos

33,5GB

10 compositores

122.709
segundos

Dataset

- John Thickstun, Zaid Harchaoui y Sham Kakade tienen su dataset de musinet en kaggle

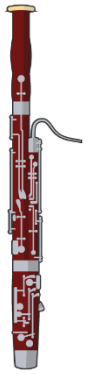

violin


viola


violonchelo


contrabajo


clarinete


fagot


oboe


bocina


flauta

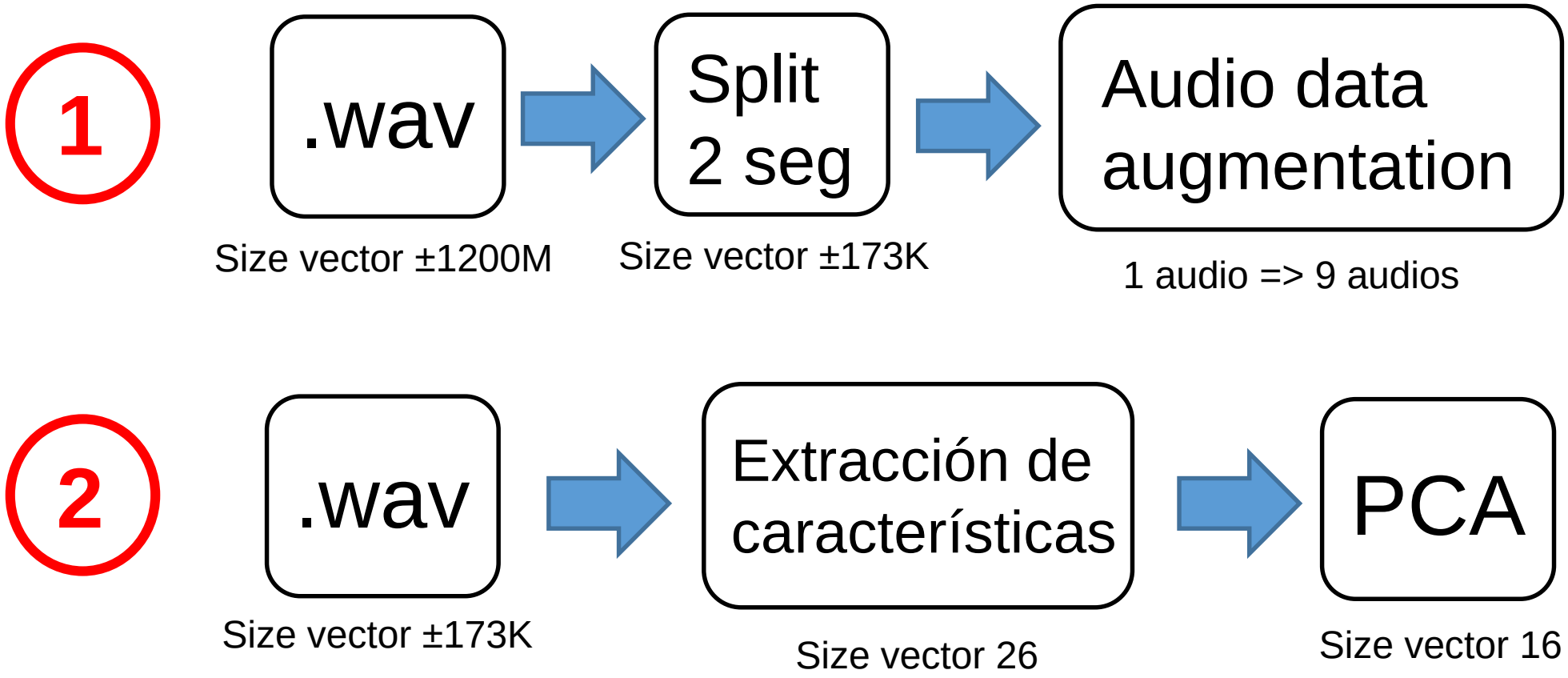

clave


piano

Preparación de los datos

Subtítulo presentación

Pre-procesado de datos



Pre-procesado de datos

Split
2 seg

Size vector ±173K

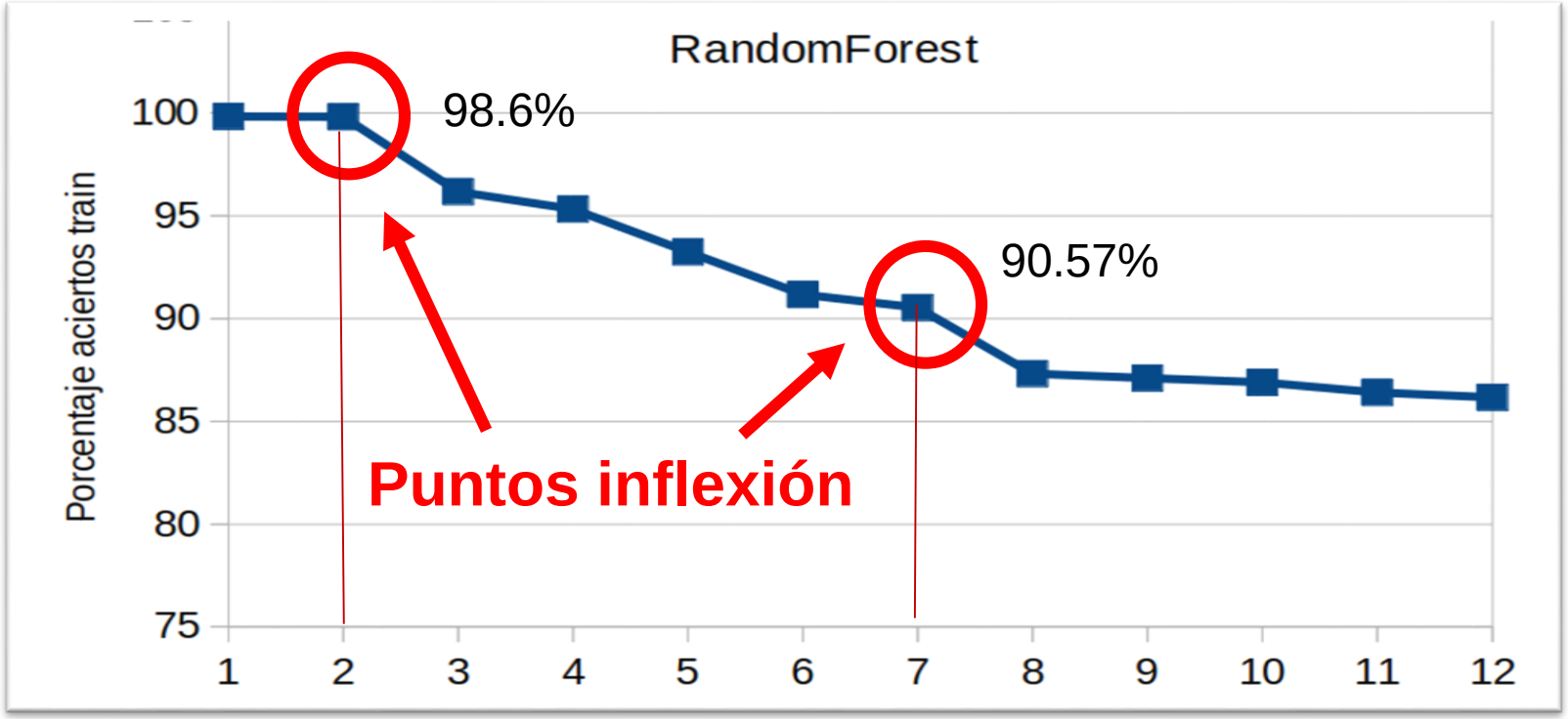


Fig 3: Resultados train vs Split segundos

```
randomforest_1seg.pkl - randomforest_2seg.pkl - randomforest_3seg.pkl - randomforest_4seg.pkl - randomforest_5seg.pkl - randomforest_6seg.pkl
randomforest_7seg.pkl - randomforest_8seg.pkl - randomforest_9seg.pkl - randomforest_10seg.pkl - randomforest_11seg.pkl - randomforest_12seg.pkl
```

Fig 4: Modelos guardados con pickle

Pre-procesado de datos

Audio data augmentation

1 audio => 9 audios

Transformaciones sobre los audios
Ruido aleatorio
Mascara para poner elementos de fondo
Pith para ajustes de tono
Ruido en base a armónicos
Volumen sobre secciones audio
Cambio de tono
Volumen sobre los audios
Cambio velocidad
Normalización de mascara
Audio original

Fig 5: Incremento de datos

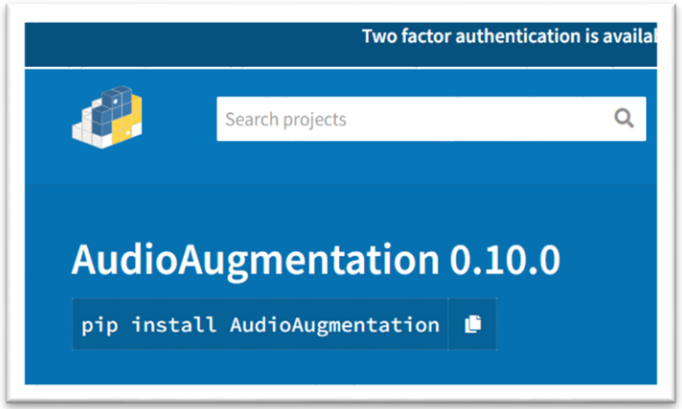


Fig 6: Librería publica

Pre-procesado de datos

Extracción de características

Size vector 26

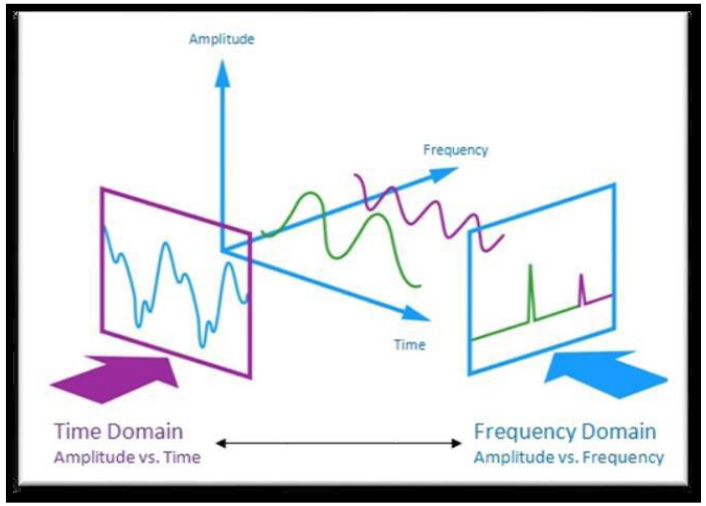


Fig 7: 3 dimensiones audio

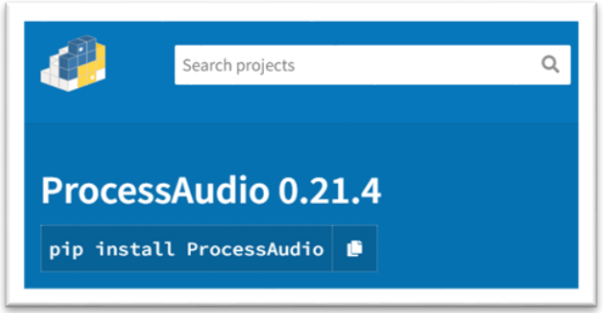


Fig 9: Librería publica

Característica	Cantidad datos
Croma	1
Rms	1
Centroide espectral	1
Ancho banda	1
Reducción espectral	1
Cruces por cero	1
MFCC	20

Fig 8: Vector características de audios

Pre-procesado de datos

PCA

Size vector 16

N componentes finales	%perdida Información
23	1
21	2
19	3
18	4
17	5
16	7
15	8
14	10

Fig 10: Distribución información

Fig 11: balanza, se busca una buena relación

Puntos elegido

Sacrifico poco pero reduzco mucho

Sacrifico el 7% pero reduzco el 42%

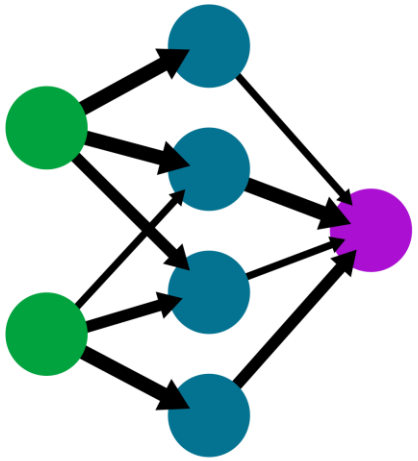
Modelamiento

Subtítulo presentación

Deep learning

- Modelado RNA simple

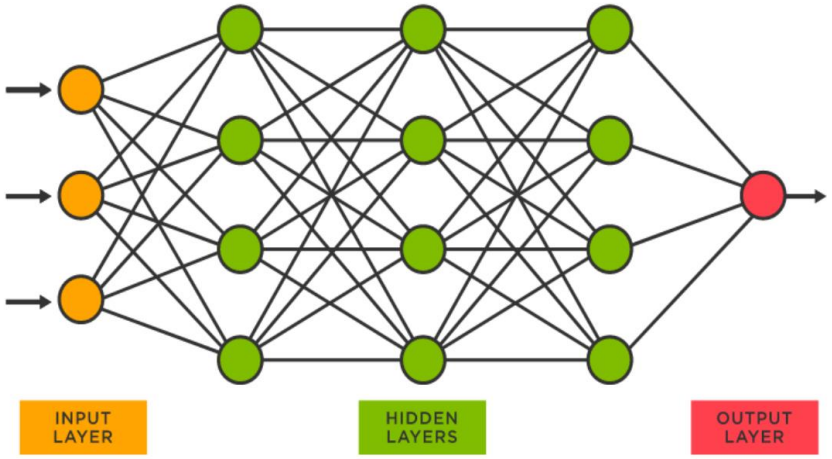
Int=16



Out=11

train	valid	test	tiempo respuesta (seg)
90,36%	90,27%	90,09%	0,06

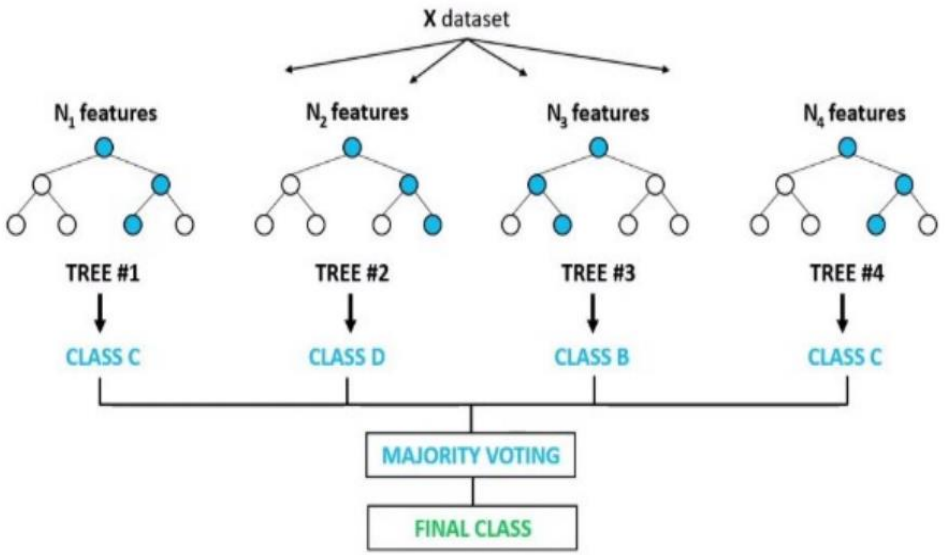
- Modelado RNA Experimental



train	valid	test	tiempo respuesta (seg)
78,63%	78,72%	80,09%	0,32

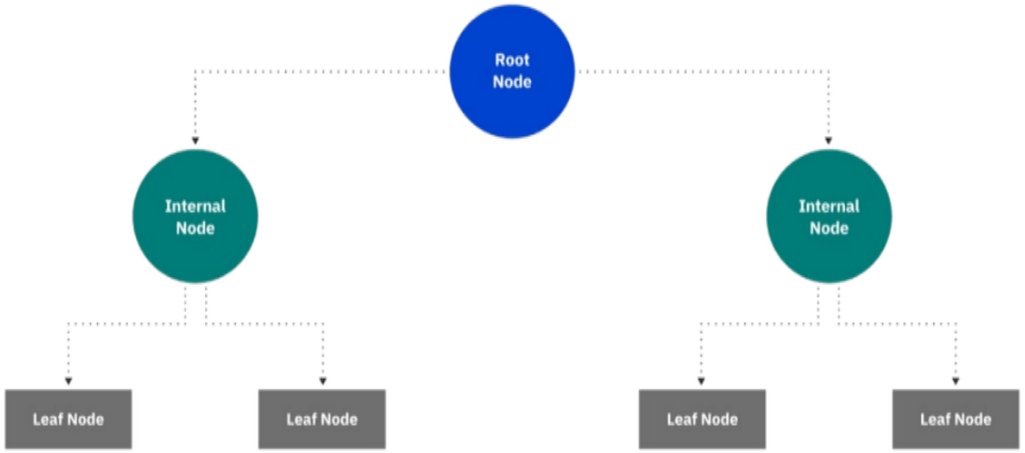
Machine learning

- Random Forest



train	valid	test	tiempo respuesta (seg)
99,81%	95,72%	94,90%	0,79

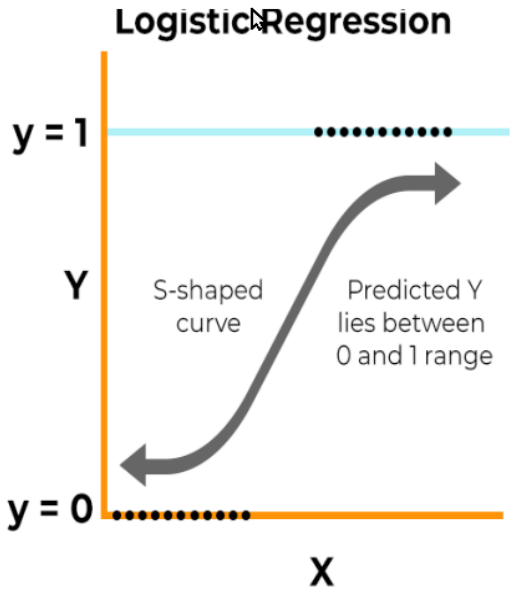
- Decision Tree



train	valid	test	tiempo respuesta (seg)
100,00%	92,54%	91,45%	1,15

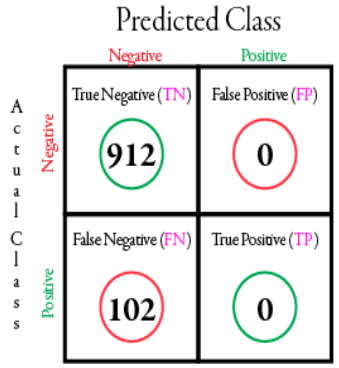
Machine learning

- Logistic Regression



train	valid	test	tiempo respuesta (seg)
92,18%	92,36%	90,72%	1,86

- Dummy Classifier



Predict negative class for every instance.

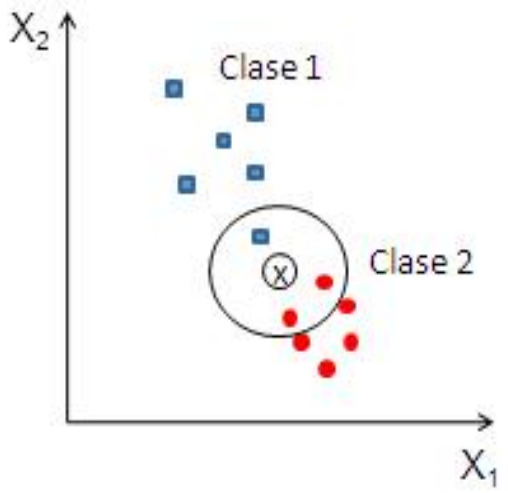
Calculate accuracy.

$$\text{Accuracy} = \frac{0 + 912}{0 + 102 + 0 + 912} = 89.94\%$$

train	valid	test	tiempo respuesta (seg)
85,54%	85,63%	81,54%	3,1

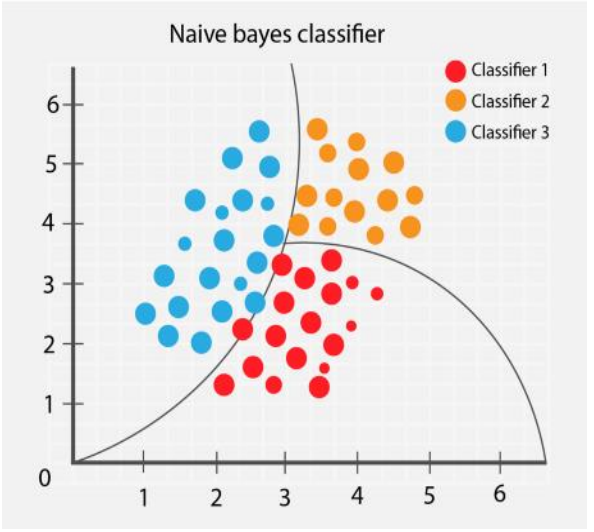
Machine learning

- Vecinos mas cercanos



train	valid	test	tiempo respuesta (seg)
96,63%	96,72%	96,00%	2,3

- Gaussian Naive Bayes



train	valid	test	tiempo respuesta (seg)
90,63%	90,81%	89,00%	1,82

Evaluación

Subtítulo presentación

Resumen

- Resumen modelado

Modelos		Algoritmo	train	valid	test	tiempo all test (seg)
Machine learning	Clásicos	RandomForestClassifier	99,81%	95,72%	94,90%	0,79
		LogisticRegression	92,18%	92,36%	90,72%	1,86
		DecisionTreeClassifier	100,00%	92,54%	91,45%	1,15
		DummyClassifier	85,54%	85,63%	81,54%	3,1
		KNeighborsClassifier	96,63%	96,72%	96,00%	2,3
		GaussianNB	90,63%	90,81%	89,00%	1,82
	Deep Learning	Basico 122.000 parametros	90,36%	90,27%	90,09%	0,06
		Experimental	78,63%	78,72%	80,09%	0,32

Mejor modelo

- Elección modelo usar

se elije el randomForest, **principalmente por sus porcentajes superiores al 95%** en validación y casi el 95% en el testeo, si bien el KNeighborsClassifier tiene mejores métricas, su tiempo de predicción es muy alto lo que lo hace ineficiente para un entorno de producción. No se elije la red neuronal básica, pues si bien otorga buenas métricas y gran velocidad de respuesta, no son tan altas las métricas como lo es con RandomForest, pero si el tema tiempo de respuesta es un factor clave se podría optar por usar la red neuronal básica).

Búsqueda hiperparametros

2952 combinaciones

```
params_grid = {
        'n_estimators': [100, 150, 200, 700],
        'max_features': ['auto', 'sqrt', 'log2'],
        'max_depth': [None] + list(range(10, 50)),
        'criterion': ['gini', 'entropy'],
        'min_samples_split': [12, 16, 20]
        }
```

```
grid = GridSearchCV(
            estimator          = rfc,
            param_grid         = params_grid,
            scoring            = 'accuracy',
            n_jobs             = multiprocessing.cpu_count() - 1,
            cv                 = KFold(
                                    n_splits=10,
                                    shuffle=True,
                                    random_state=seed
                                 ),
            refit              = True,
            verbose            = 10,
            return_train_score = True
            )
```

19 días

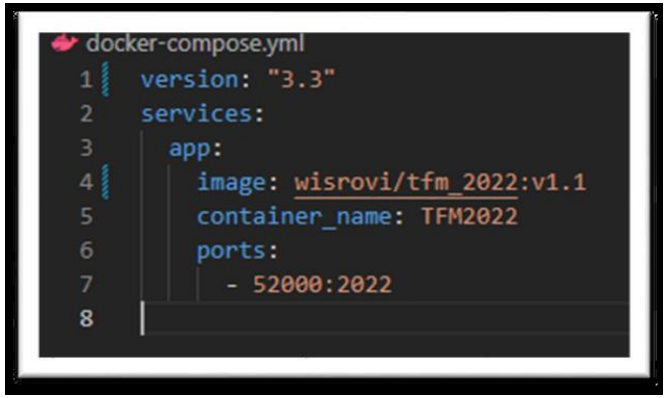
```
rfc = RandomForestClassifier(n_jobs=-1,
                             max_features= 'sqrt' ,
                             n_estimators=50,
                             verbose=0,
                             oob_score = True)
```

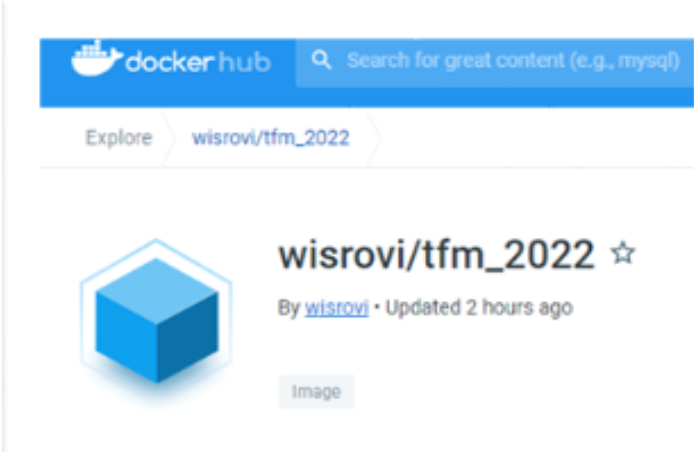
```
rfc = RandomForestClassifier(n_jobs=-1,
                                    max_features      = 'sqrt' ,
                                    n_estimators      = 200,
                                    max_depth         = 75,
                                    criterion         = 'gini',
                                    min_samples_split = 20,
                                    verbose           = 0,
                                    oob_score         = True
                                )
```

Implementación

Subtítulo presentación

Resultados finales

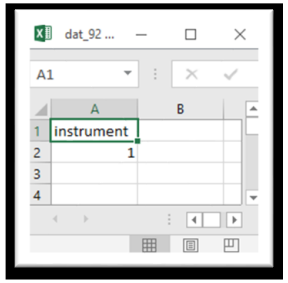




Subtítulo presentación

Resultados finales

dat_92.wav



Subtítulo presentación

API versión 1.0

http

Flask
RESTful API
GET PUT POST DELETE

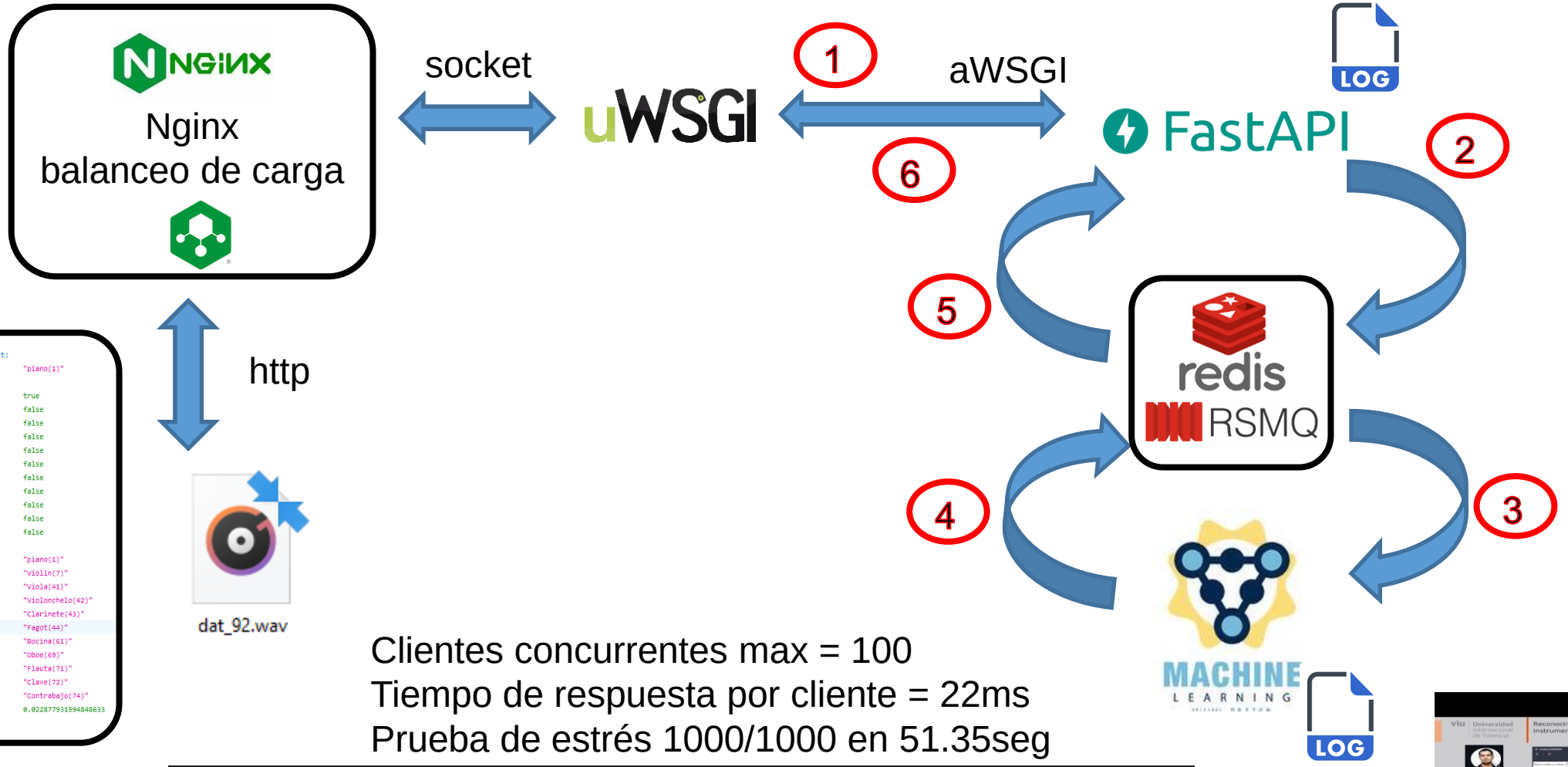
MACHINE
LEARNING

1 proceso síncrono

Clientes concurrentes max = 3
Tiempo de respuesta por cliente = 130ms

API versión 2.0

Multiprocesos asíncronos



Clientes concurrentes max = 100
Tiempo de respuesta por cliente = 22ms
Prueba de estrés 1000/1000 en 51.35seg

```
2022-10-22 05:21:22 INFO     Total test: 1000 - tiempo: 51.348 -  Resultado: OK: 1000
 / BAD: 0
```

Subtítulo presentación

Trabajos futuros

Subtítulo presentación

Mejoras

- Entrenar un modelo que reciba audios de diferente medida de tiempo para hacer mas robusto el modelo.
- Continuar con las RNA pues son mas rápidos y fácil de implantar en producción, incluso TFlite, además de que podría usarse para transfer learning para trabajos similares.
- Buscar mejores hiperparametros.
- Unificar las librerías creadas.
- Lograr implantar la API en la raspberry, Jetson o similar para trabajar el modelo en edge computing para acelerar la predicción y mantener la confidencialidad de los datos.

Conclusiones

Subtítulo presentación

Conclusiones

- Las redes neuronales no siempre son la mejor opción para todos los problemas.
- Limpiar y preparar los datos es sumamente importante
- El PCA ayuda considerablemente a mejorar el aprendizaje del modelo
- Los algoritmos clásicos dieron mejor resultado de lo esperado permitiendo dar continuidad a trabajos relacionados.

Gracias