Setting the Standard for Automation™

ISA Sección Española

Bloque 2: Técnicas avanzadas

j.ordieres@upm.es

10/05/2017

Estándares
Certificaciones
Formación
Publicaciones
Conferencias

INDICE

- Caso de ejemplo
- Metodología
- Caso de trabajo
  - Preprocesado
  - Selección
  - Modelado
  - Validación
- Conclusión

Caso de ejemplo

SIDERURGIA

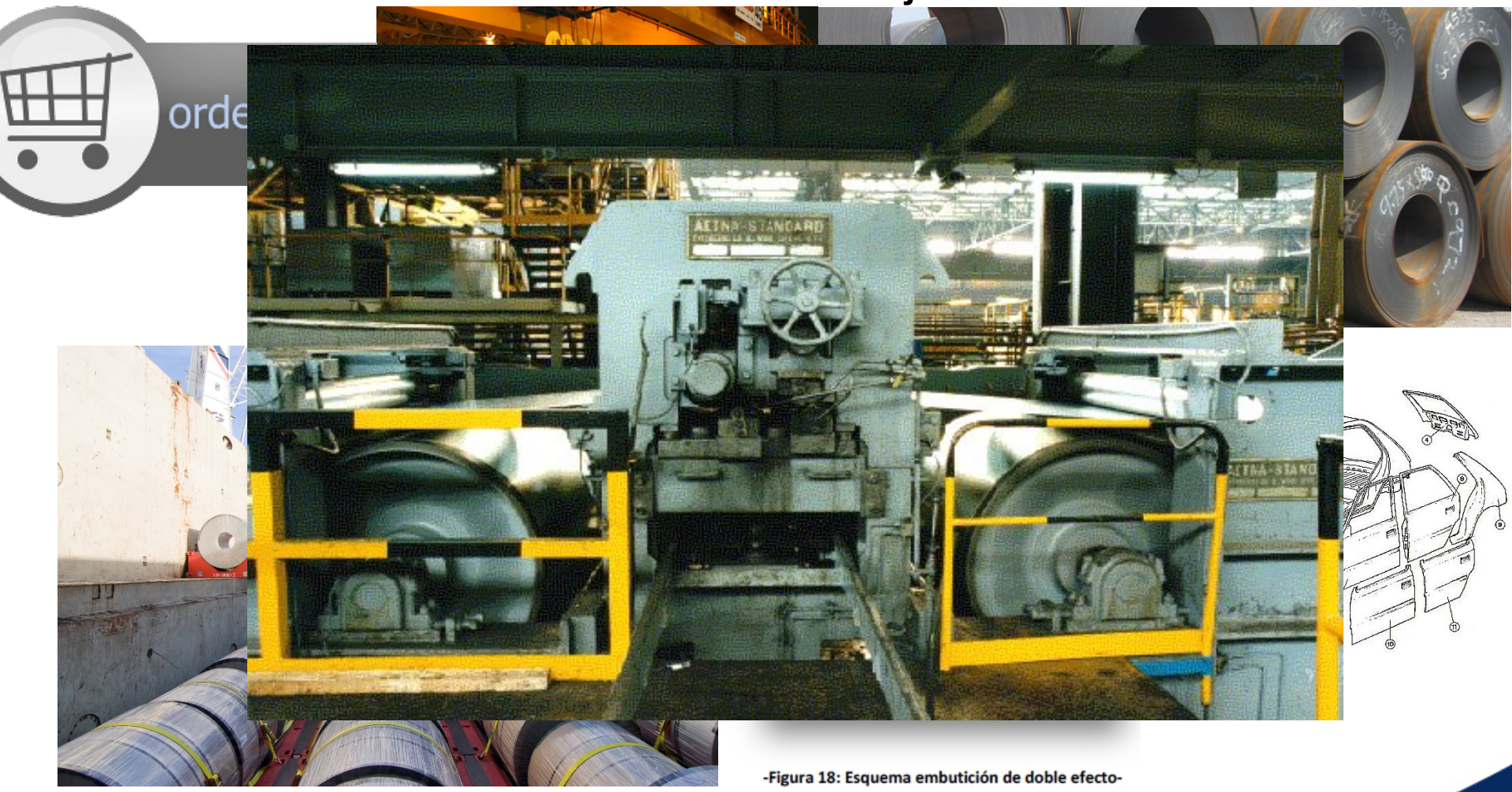
Varios ámbitos de aplicación:

- Calidad (*Determinar el reemplazo de cilindros según defectos)*

- Setup en procesos con control en bucle abierto (*control de espesor de bobina en semicontinuo*)

Caso de ejemplo

SIDERURGIA => El caso del “cerrojo”:



-Figura 18: Esquema embutición de doble efecto-

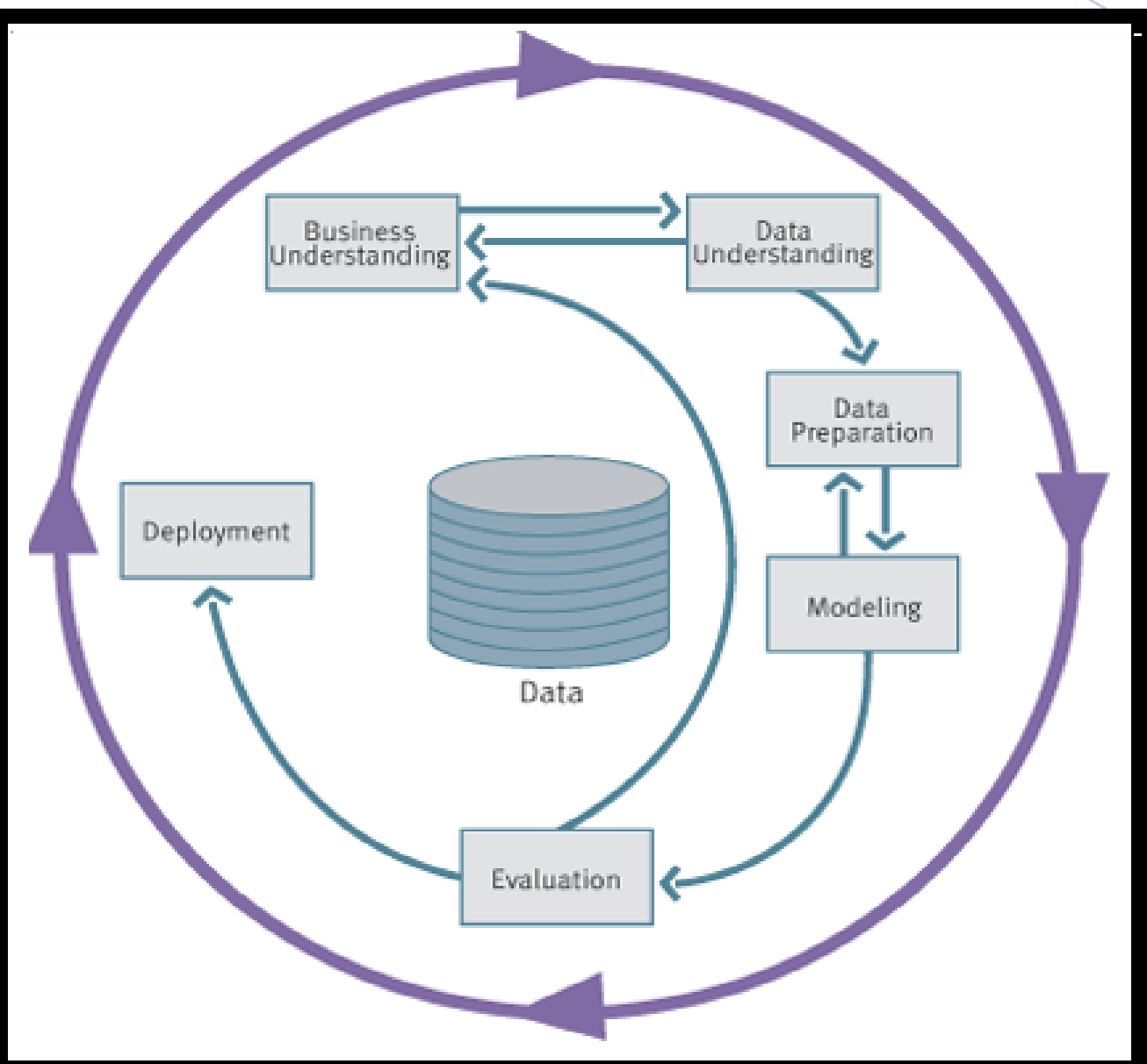
Metodología

CRISP/DM
Varios ámbitos de aplicación =>

Varias metodologías:

- CRISP/DM,
- Fayads,
- SEMMA,

etc.

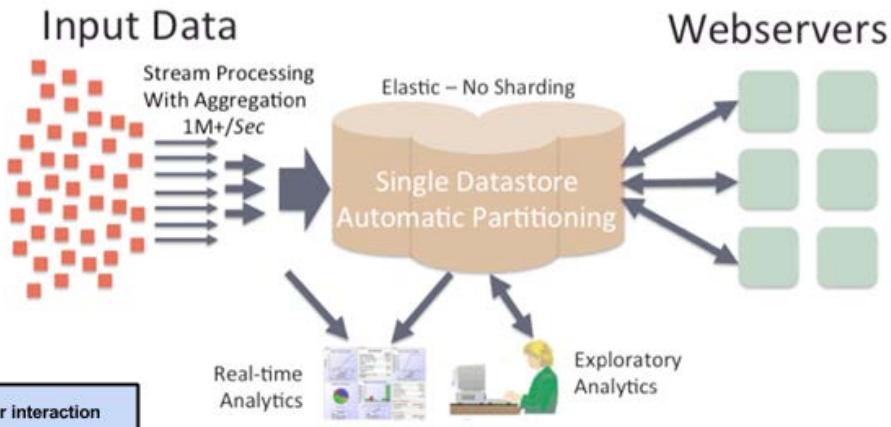


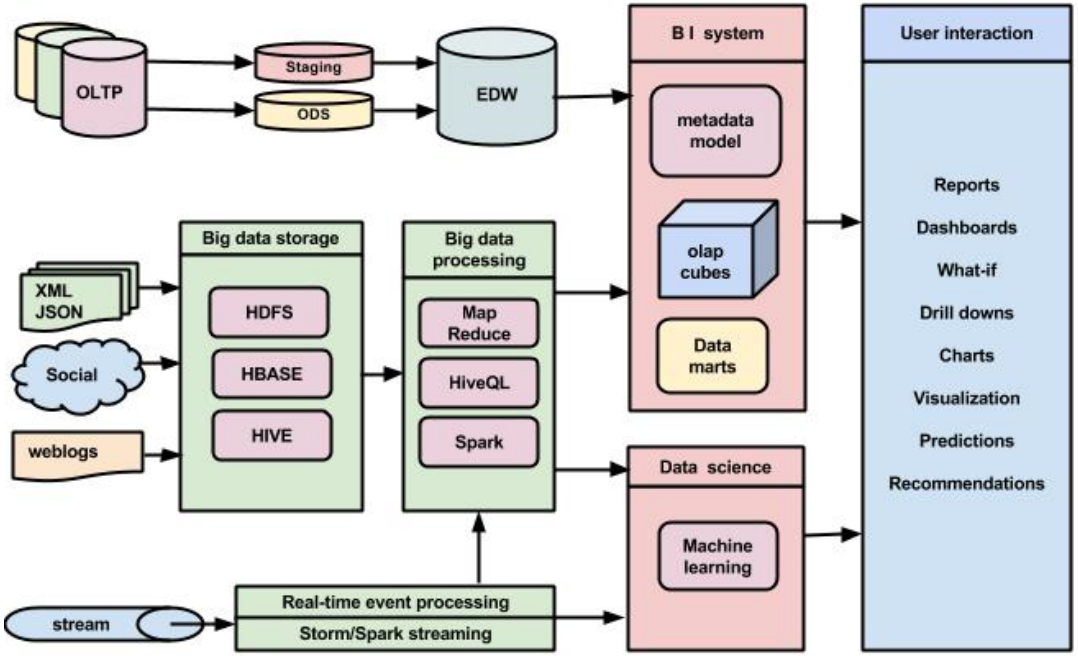
Metodología

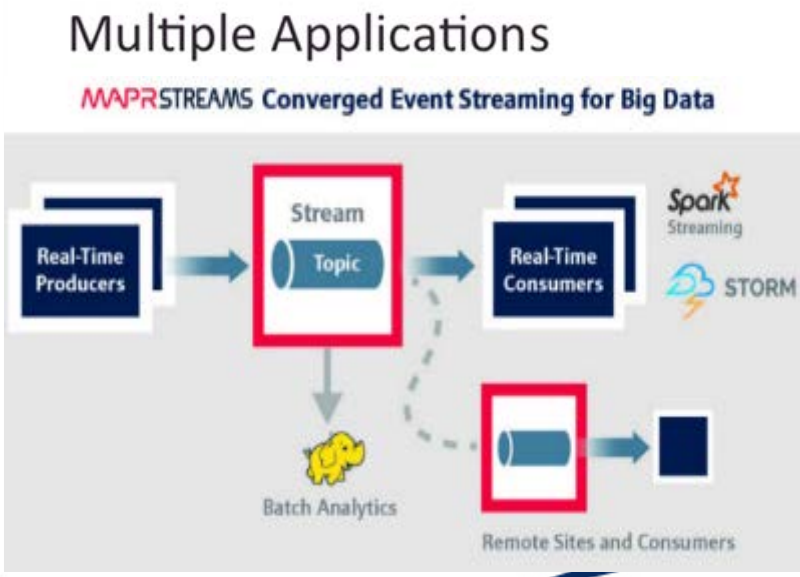
CRISP/DM / Data Preparation

Incluye:

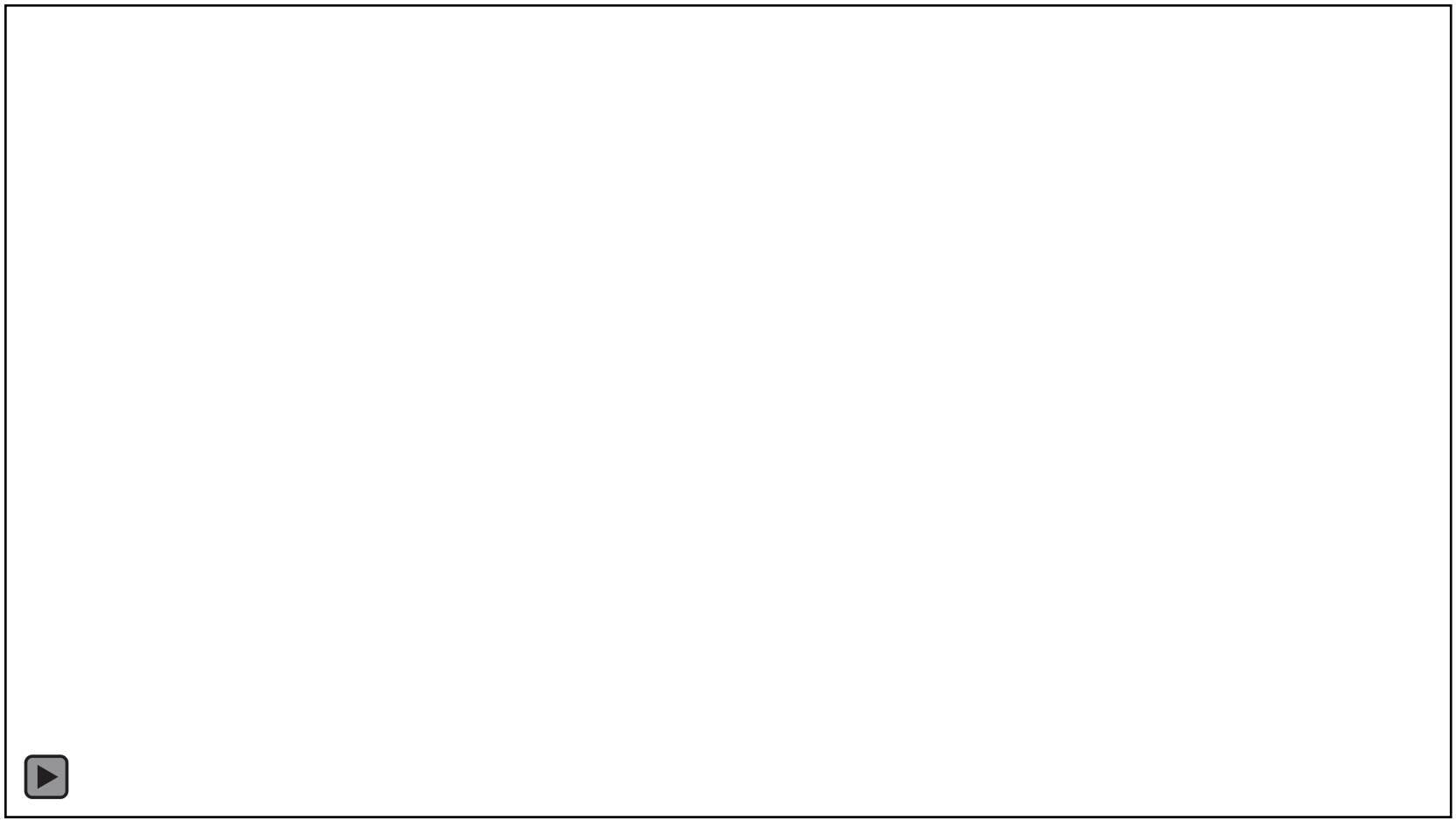
* “Data ingestion”
* “Data preprocessing”
* “Outlier identification”
* “Data transformation”
* “Data storage”







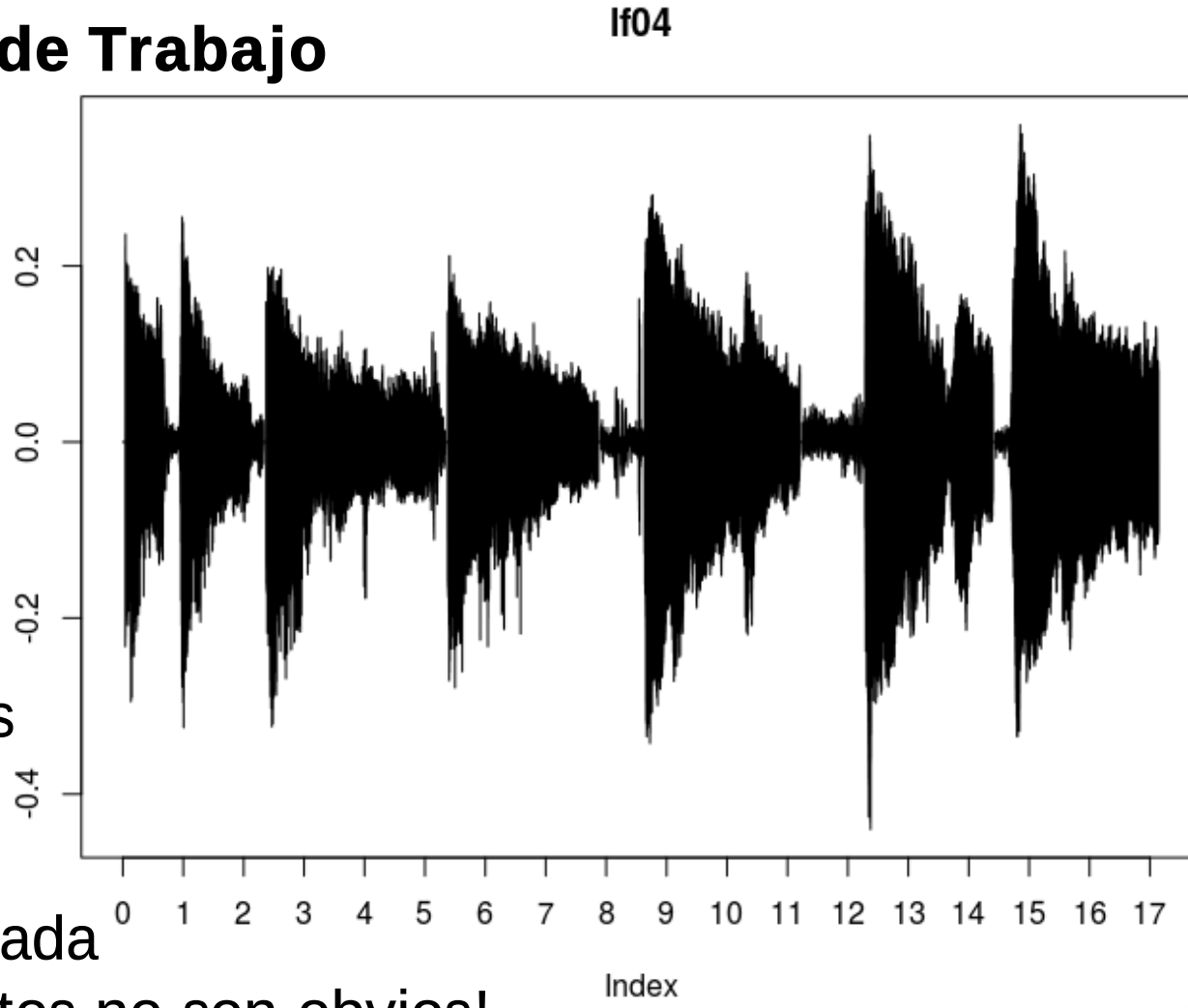
CRISP/DM / Caso de Trabajo



Caso de Trabajo

CRISP/DM / Caso de Trabajo

Incluye:



Dataset:

- Voces con Parkinson
- Voces sin Parkinson

Varias decenas de casos

Tipo de Problema:

- Clasificación supervisada
- Dificultad: ¡los atributos no son obvios!

Caso de Trabajo

CRISP/DM / Preprocesado

```
#
library(tuneR)
library(signal)
```

```
##
## Attaching package: 'signal'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, poly
```

```
FunFrequencyPlot<-function(l,f,p,main){
  plot.frequency.spectrum (fft(l),
              xlim=c(0,length(l))/p,ylim=c(0,0.002),
              n=length(l),main = main)
  axis(1, at=seq(0,length(l)/2,2000*length(l)/f),
       labels=seq(0,f/(2*1000),2))
}
#
#Function to produce the plots
FunPlotAudio<-function(data, label, f, path){
  for(i in 1:length(data)){
    s<-data[[i]]
    fi<-f[i] #sampling frequency
    spx<-specgram(s, n=512, Fs=fi, overlap = 400)
    fname<-paste(path,i,"_",label[i],".png",sep = "")
    png(fname,width=288,height = 288)
    par(mar=c(0, 0, 0, 0), xaxs='i', yaxs='i')
    plot(spx, col = heat.colors(7, alpha = 1))
    dev.off()
  }
}
#
#http://www.di.fc.ul.pt/~jpn/r/fourier/fourier.html
#X.K is the result from fft() operation to a vector
```

```
#http://www.di.fc.ul.pt/~jpn/r/fourier/fourier.html
#X.K is the result from fft() operation to a vector
#n is the number of samples
plot.frequency.spectrum <- function(X.k, xlim,ylim, main="", n) {
  # ylimits=c(0,max(Mod(X.k)[-1]))
  #ylimits=c(0,max(Mod(X.k)[-1])*2/n)
  xlimits=xlim
  ylimits=ylim
  plot.data  <- cbind(0:(length(X.k)-1), Mod(X.k))
  plot.data[2:length(X.k),2] <- 2*plot.data[2:length(X.k),2]/n
  plot(plot.data, type="h", lwd=1, main=main,
       xlab="Frequency (kHz)", ylab="Amplitude",
       xlim=xlimits, ylim=ylimits,xaxt="n")
}
#
###Function to use a sliding window to chunk the data list
# win<-65536 #window size of to filter the chunk
# step<-4096 #moving step of the window
# lab #label list,same length with df
# lst  #raw data list to be processed
# f # sampling frequency
# sound # list of sound "a""o"
# subject #name of subject
```

Caso de Trabajo

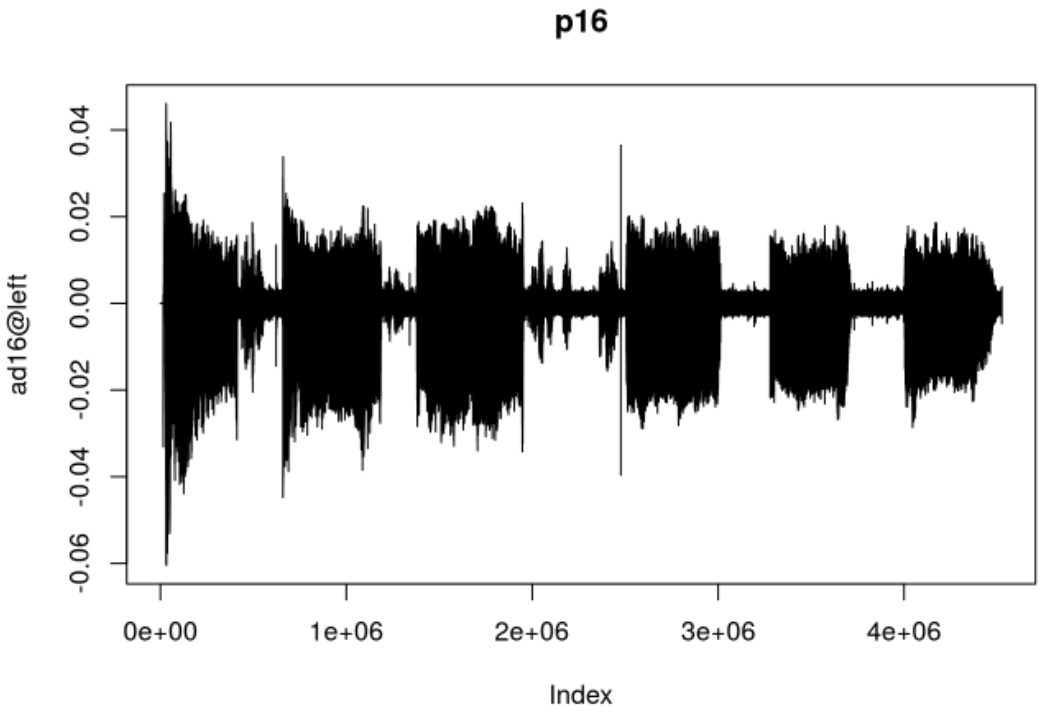
CRISP/DM / Preprocesado

```
FunChunkAudio<-function (lst, lab, win,step, f,sound, subject){
  data<-NULL #List to hold the raw data
  label<-NULL
  freq<-NULL
  snd<-NULL
  sub<-NULL
  l<-length(lst)
  for(i in 1:l){
    li<-lst[[i]] #the i vector
    labi<-lab[i] #the label of i vector
    fi<-f[i]
    soundi<-sound[i]
    subjecti<-subject[i]
    if (length(li)>=win){
      n <- floor((length(li)-win)/step)+1 #number of chunks
      for(j in 1:n){
        s0<-(j-1)*step+1 #start of one chunk
        s1<-s0+win-1
        dj<-li[s0:s1]
        if(is.null(data)){
          data<-list(dj)
          label<-labi
          freq<-fi
          snd<-soundi
          sub<-subjecti
        } else {
          k<-length(data)
          data[[k+1]]<-dj
          label<-c(label,labi)
          freq<-c(freq,fi)
          snd<-c(snd,soundi)
          sub<-c(sub,subjecti)
        }
      }
    }
  }
  lablist<-list(data,label,freq,snd,sub)
  return(lablist)
}
```

Lectura de muestras

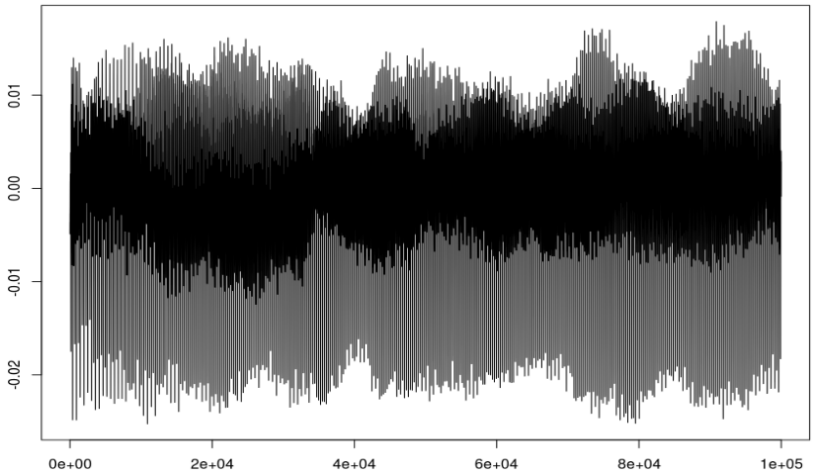
Check the general characters

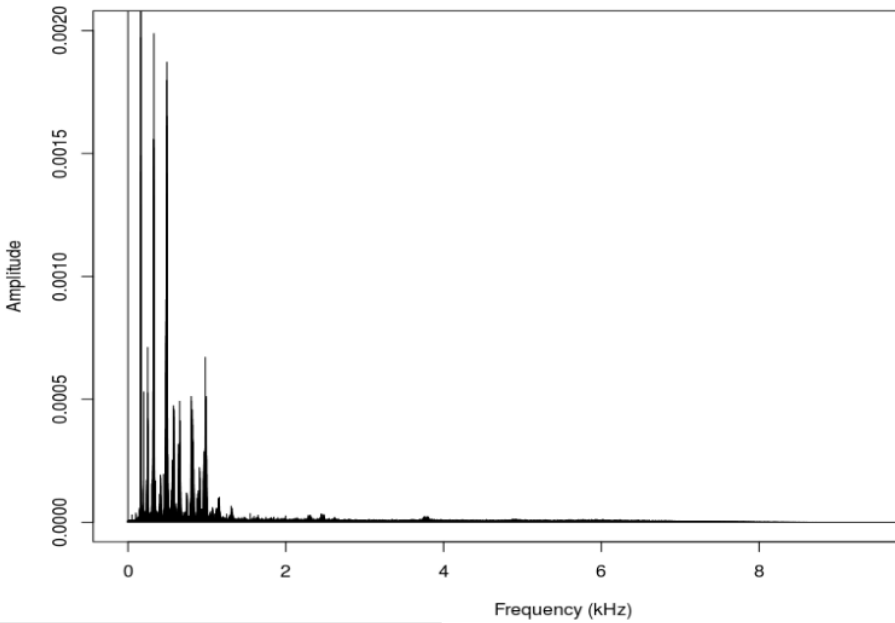
```
###Patient16
setwd('/home/jb/git/DeepLearning/pdaudio')
ad16<-readWave("audio/016.wav")
f16<-ad16@samp.rate
plot(ad16@left,type="l",main="p16")
```

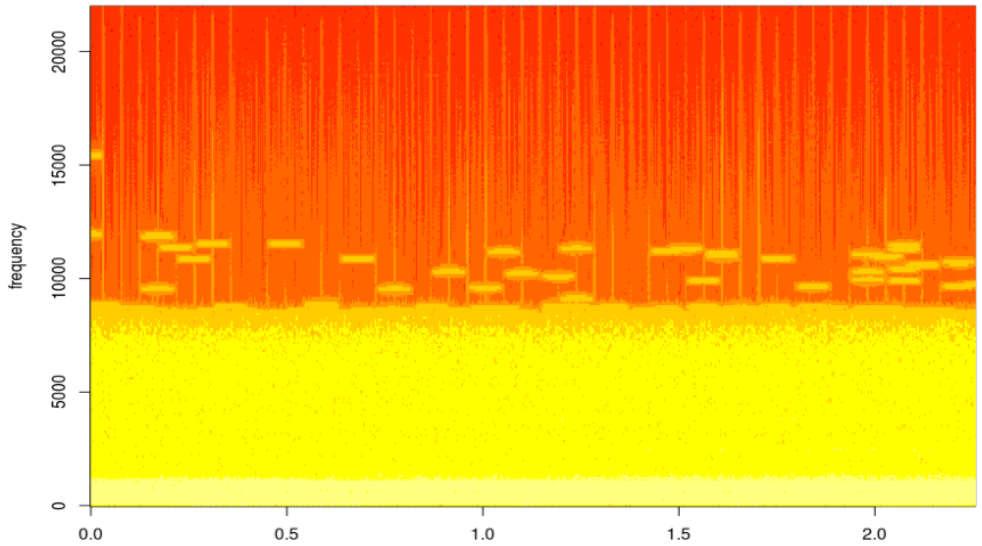


Caso de Trabajo

CRISP/DM / Selección







Caso de Trabajo

CRISP/DM / Selección

Preprocess the data, manually

```
####CHUNK DATASETS
##Change the path to your own
ad2<-readWave("audio/002.wav")
ad9<-readWave("audio/009.wav")
ad16<-readWave("audio/016.wav")
adzh1<-readWave("audio/zh.wav")
adzh2<-readWave("audio/xc.wav")
adjose<-readWave("audio/jose.wav")

lf2<-ad2@left
lf9<-ad9@left
lf16<-ad16@left
lfzh1<-adzh1@left
lfzh2<-adzh2@left
lfjose<-adjose@left

##manually cut
# lf2.a<-lf2[c(25000:130000,210000:310000,425000:515000)]
# lf2.o<-lf2[c(640000:720000, 820000:900000, 1045000:1135000)]
# lf9.a<-lf9[c(30000:180000,340000:460000,640000:820000)]
# lf9.o<-lf9[c(900000:1000000,1090000:1200000,1260000:1380000)]
# lf16.a<-lf16[c(100000:400000,660000:1190000,1400000:1940000)]
# lf16.o<-lf16[c(2510000:3000000,3280000:3700000,4000000:4450000)]
# lfzh1.a<-lfzh1[c(100000:350000, 500000:700000, 820000:980000 )]
# lfzh1.o<-lfzh1[c(1110000:1240000, 1340000:1470000, 1610000:1760000)]
# lfzh2.a<-lfzh2[c(180000:360000,510000:690000, 870000:1000000)]
# lfzh2.o<-lfzh2[c(1150000:1260000, 1380000:1490000,1600000:1730000 )]
# lfjose.a<-lfjose[c(40000:300000,900000:1150000, 1620000:1850000)]
# lfjose.o<-lfjose[c(450000:790000,1280000:1530000,1980000:2130000)]

lf.ao<-list(lf2[25000:130000],lf2[210000:310000],lf2[425000:515000],
          lf9[30000:180000],lf9[340000:460000], lf9[640000:820000],
          lf16[100000:400000],lf16[660000:1190000],lf16[1400000:1940000],
          lfzh1[100000:350000],lfzh1[500000:700000],lfzh1[820000:980000],
          lfzh2[180000:360000],lfzh2[510000:690000],lfzh2[870000:1000000],
          lfjose[40000:300000],lfjose[900000:1150000],lfjose[1620000:1850000],
          lf2[640000:720000],lf2[820000:900000], lf2[1045000:1135000],
          lf9[900000:1000000],lf9[1090000:1200000], lf9[1260000:1380000],
          lf16[2510000:3000000],lf16[3280000:3700000],lf16[4000000:4450000],
          lfzh1[1110000:1240000],lfzh1[1340000:1470000],lfzh1[1610000:1760000],
          lfzh2[1150000:1260000],lfzh2[1380000:1490000],lfzh2[1600000:1730000],
          lfjose[450000:790000],lfjose[1280000:1530000],lfjose[1980000:2130000])
lab.sound<-rep(c(1,0),each=18) #"a"==1, "o"==0
lab.pd<-rep(c(1,0,1,0),each=9)
lab.sub<-rep(rep(c("p2","p9","p16","zh1","zh2","jose"),each=3),2)
lab.f<-rep(c(44100,48000,44100,48000),each=9)
#                  data    label  win   step  frequency  "a""o"  subject
lablist<-FunChunkAudio(lf.ao, lab.pd, 32768, 2048, lab.f,  lab.sound, lab.sub)
ck.data<-lablist[[1]]
ck.label<-lablist[[2]]
ck.f<-lablist[[3]]
```

The above code will produce a list containing data chunks with length 32768 each, and followed by the label and sampling frequency etc.

```
for(i in 1:length(ck.data)){
   ck.data[[i]][is.na(ck.data[[i]])]<-0
}
test.id<-sample(1:length(ck.data),200)
test.data<-ck.data[test.id]
test.label<-ck.label[test.id]
test.f<-ck.f[test.id]
train.data<-ck.data[-test.id]
train.label<-ck.label[-test.id]
train.f<-ck.f[-test.id]


#Produce spectrgram plots and save them to local folders
path_train = "PD_train/"
path_test = "PD_test/"
FunPlotAudio(test.data, test.label, test.f, path_test)
```

1986_1.png	6.2 KB	Apr 22, 2017, 7:05 PM	
1987_1.png	6.2 KB	Apr 22, 2017, 7:05 PM	
1988_1.png	6.1 KB	Apr 22, 2017, 7:05 PM	
1989_1.png	6.2 KB	Apr 22, 2017, 7:05 PM	
198_1.png	6.3 KB	Apr 22, 2017, 7:01 PM	
1990_1.png	6 KB	Apr 22, 2017, 7:05 PM	
1991_1.png	6.5 KB	Apr 22, 2017, 7:05 PM	
1992_1.png	6.1 KB	Apr 22, 2017, 7:05 PM	
1993_1.png	6.2 KB	Apr 22, 2017, 7:05 PM	
1994_1.png	6.1 KB	Apr 22, 2017, 7:05 PM	
1995_1.png	6.9 KB	Apr 22, 2017, 7:05 PM	
1996_1.png	6.2 KB	Apr 22, 2017, 7:05 PM	
1997_1.png	6.3 KB	Apr 22, 2017, 7:05 PM	
1998_1.png	6.1 KB	Apr 22, 2017, 7:05 PM	
1999_1.png	6 KB	Apr 22, 2017, 7:05 PM	
199_1.png	6.2 KB	Apr 22, 2017, 7:01 PM	
19_1.png	7.6 KB	Apr 22, 2017, 7:00 PM	
1_1.png	8.4 KB	Apr 22, 2017, 7:00 PM	
2000_1.png	6.2 KB	Apr 22, 2017, 7:05 PM	
2001_1.png	6.9 KB	Apr 22, 2017, 7:05 PM	
2002_1.png	6 KB	Apr 22, 2017, 7:05 PM	
2003_1.png	6.7 KB	Apr 22, 2017, 7:05 PM	
2004_1.png	6.3 KB	Apr 22, 2017, 7:05 PM	
2005_1.png	6.2 KB	Apr 22, 2017, 7:05 PM	
2006_1.png	6 KB	Apr 22, 2017, 7:05 PM	

..			
100_1.png	7.6 KB	Apr 22, 2017, 7:00 PM	
101_1.png	6.3 KB	Apr 22, 2017, 7:00 PM	
102_0.png	8.8 KB	Apr 22, 2017, 7:00 PM	
103_1.png	7.2 KB	Apr 22, 2017, 7:00 PM	
104_1.png	6.6 KB	Apr 22, 2017, 7:00 PM	
105_0.png	8.3 KB	Apr 22, 2017, 7:00 PM	
106_0.png	8.1 KB	Apr 22, 2017, 7:00 PM	
107_0.png	8.1 KB	Apr 22, 2017, 7:00 PM	
108_1.png	6.6 KB	Apr 22, 2017, 7:00 PM	
109_1.png	7.1 KB	Apr 22, 2017, 7:00 PM	
10_1.png	6.7 KB	Apr 22, 2017, 7:00 PM	
110_0.png	7.6 KB	Apr 22, 2017, 7:00 PM	
111_0.png	730 B	Apr 22, 2017, 7:00 PM	
112_1.png	6.7 KB	Apr 22, 2017, 7:00 PM	
113_0.png	3.8 KB	Apr 22, 2017, 7:00 PM	
114_1.png	7.4 KB	Apr 22, 2017, 7:00 PM	
115_0.png	8.4 KB	Apr 22, 2017, 7:00 PM	
116_0.png	7.3 KB	Apr 22, 2017, 7:00 PM	
117_0.png	8.1 KB	Apr 22, 2017, 7:00 PM	
118_1.png	6.8 KB	Apr 22, 2017, 7:00 PM	
119_1.png	6.1 KB	Apr 22, 2017, 7:00 PM	
11_1.png	6.7 KB	Apr 22, 2017, 7:00 PM	
120_0.png	7.4 KB	Apr 22, 2017, 7:00 PM	

Caso de Trabajo

CRISP/DM / Selección y Modelado

jupyter Untitled Last Checkpoint: a few seconds ago (autosaved)

Control Panel Logout

File Edit View Insert Cell Kernel Help Python 3

Code CellToolbar

Carga de datos de las imágenes creadas en R

In [1]:
```
import cv2
import numpy as np
import os
from random import shuffle
from tqdm import tqdm
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression
import tensorflow as tf
import matplotlib.pyplot as plt
```

Después de cargar las librerías de python necesarias para el preprocesado de los objetos, denifimos variables de configuración

In [2]:
```
TRAIN_DIR = '/home/jb/git/DeepLearning/pdaudio/PD_train'
TEST_DIR = '/home/jb/git/DeepLearning/pdaudio/PD_test'
IMG_SIZE = 100
LR = 1e-3
MODEL_NAME = 'twoclass-{}-{}.model'.format(LR, '2conv-basic')
```

Ahora creamos algunas funciones de utilidad para poder leer las imágenes y de su nombre la clase a la que pertenecen.

Caso de Trabajo

CRISP/DM / Selección y Modelado

jupyter Untitled Last Checkpoint: a few seconds ago (autosaved)

Control Panel Logout

File Edit View Insert Cell Kernel Help Python 3

Code CellToolbar

Carga de datos de las imágenes creadas en R

In [1]:
```
import cv2
import numpy as np
import os
from random import shuffle
from tqdm import tqdm
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression
import tensorflow as tf
import matplotlib.pyplot as plt
```

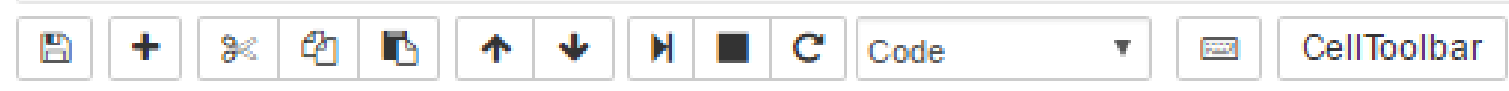
Después de cargar las librerías de python necesarias para el preprocesado de los objetos, denifimos variables de configuración

In [2]:
```
TRAIN_DIR = '/home/jb/git/DeepLearning/pdaudio/PD_train'
TEST_DIR = '/home/jb/git/DeepLearning/pdaudio/PD_test'
IMG_SIZE = 100
LR = 1e-3
MODEL_NAME = 'twoclass-{}-{}.model'.format(LR, '2conv-basic')
```

Ahora creamos algunas funciones de utilidad para poder leer las imágenes y de su nombre la clase a la que pertenecen.

Caso de Trabajo

CRISP/DM / Selección y Modelado




```
    shuffle(testing_data)
    np.save('test_data.npy', testing_data)
    return testing_data
```

Procesado de los datos

Cargamos las imágenes y sus etiquetas.

```
In [4]: train_data = create_train_data()
```

```
100%|██████████| 2853/2853 [00:17<00:00, 164.36it/s]
```

CRISP/DM / Modelado

Definición de la DNN

Comenzamos a construir la **red neuronal convolutiva** (CNN). El primer paso la definición de la arquitectura

In [5]:
```
tf.reset_default_graph()
# Comenzamos con la capa de entrada
convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 1], name='input')
# Hacemos una convolución de imagenes 5*5 con 32 capas de profundidad
convnet = conv_2d(convnet, 32, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)
# Hacemos una convolución de imagenes 5*5 con 64 capas de profundidad
convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)
# Hacemos una convolución de imagenes 5*5 con 128 capas de profundidad
convnet = conv_2d(convnet, 128, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)
# Hacemos una convolución de imagenes 5*5 con 64 capas de profundidad
convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)
# Hacemos una convolución de imagenes 5*5 con 32 capas de profundidad
convnet = conv_2d(convnet, 32, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)
# Ahora la red neural convencional ya sin convoluciones pero con activación tipo imagen
convnet = fully_connected(convnet, 1024, activation='relu')
convnet = dropout(convnet, 0.8)
# Ahora la red neural convencional ya para efectos sigmoidales
convnet = fully_connected(convnet, 2, activation='softmax')
#
# Ahora construimos el regresoar dadas las features creadas
convnet = regression(convnet, optimizer='adam', learning_rate=LR,
                     loss='categorical_crossentropy', name='targets')

model = tflearn.DNN(convnet, tensorboard_dir='log')
```

Caso de Trabajo

CRISP/DM / Modelado

jupyter Untitled Last Checkpoint: 7 minutes ago (autosaved)

File Edit View Insert Cell Kernel Help

Markdown CellToolbar

Ahora nos preocupamos de cargar el modelo si existe

In [6]:
```
if os.path.exists('{}.meta'.format(MODEL_NAME)):
    model.load(MODEL_NAME)
    print('model loaded!')
```

Vamos a preparar ahora los datos para el entrenamiento. Respecto de las imágenes con etiqueta existentes, y para entrenar y las 200 últimas para validar el aprendizaje

In [7]:
```
print(len(train_data))
train = train_data[:-200]
test = train_data[-200:]
```

```
2853
```

CRISP/DM / Modelado

Entrenamiento de la CNN

In [8]:
```
X = np.array([i[0] for i in train]).reshape(-1,IMG_SIZE,IMG_SIZE,1)
Y = [i[1] for i in train]

test_x = np.array([i[0] for i in test]).reshape(-1,IMG_SIZE,IMG_SIZE,1)
test_y = [i[1] for i in test]
```

Ahora entrenamos el modelo

In [9]:
```
model.fit({'input': X}, {'targets': Y}, n_epoch=5,
          validation_set=({'input': test_x}, {'targets': test_y}),
          snapshot_step=500, show_metric=True, run_id=MODEL_NAME)
```

```
Training Step: 209  | total loss: 0.04912 | time: 23.736s
| Adam | epoch: 005 | loss: 0.04912 - acc: 0.9967 -- iter: 2624/2653
Training Step: 210  | total loss: 0.04421 | time: 25.311s
| Adam | epoch: 005 | loss: 0.04421 - acc: 0.9971 | val_loss: 0.00001 - val_acc: 1.0000 -- iter: 2653/2653
--
```

CRISP/DM / Validación

Validación de la red

Ahora preparamos las imágenes que no han sido vistas antes durante la creación del modelo

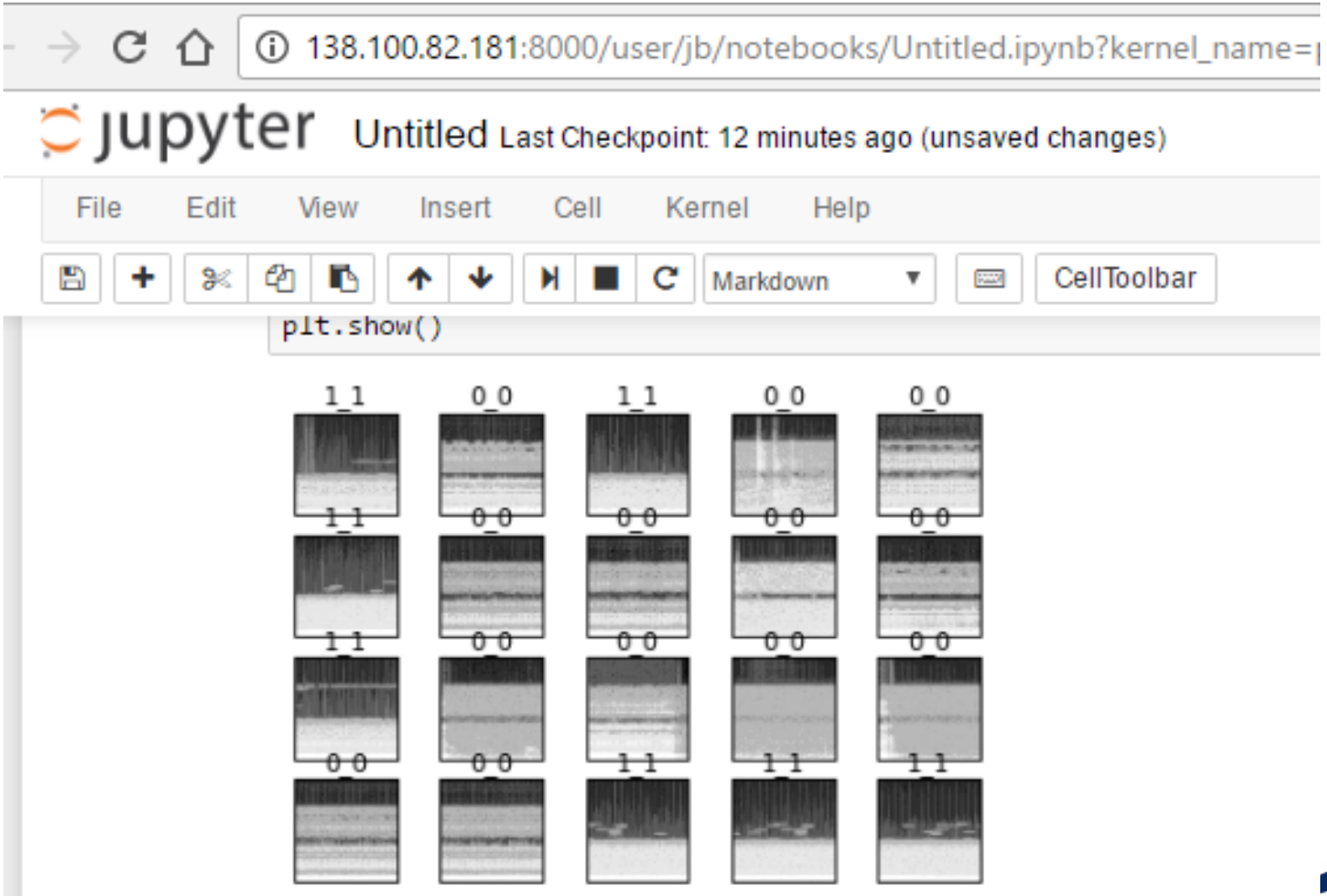
In [10]:
```
test_data = process_test_data()
```

```
100%|██████████| 200/200 [00:00<00:00, 459.18it/s]
```

In [11]:
```
fig=plt.figure()
#
for num,data in enumerate(test_data[:20]):
    img_num = data[1]
    img_data = data[0]

    y = fig.add_subplot(4,5,num+1)
    orig = img_data
    data = img_data.reshape(IMG_SIZE,IMG_SIZE,1)
    #model_out = model.predict([data])[0]
    model_out = model.predict([data])[0]

    if np.argmax(model_out) == 1: str_label='1'
    else: str_label='0'

    str_label=str_label+"_"+img_num
    y.imshow(orig,cmap='gray')
    plt.title(str_label)
    y.axes.get_xaxis().set_visible(False)
    y.axes.get_yaxis().set_visible(False)

plt.show()
```

Caso de Trabajo

CRISP/DM / Validación



Conclusión

- Hemos visto la puesta en práctica de modelización basada en datos.
- En la sesión de ayer con técnicas convencionales se ha visto técnicas de clasificación
- Hoy hemos visto como construir esos modelos cuando no es obvia la identificación de features y cuando hablamos de patrones temporales.
- Ello conlleva un preprocesado intensivo (espectrogramas, etc.) y un trabajo convolutivo de cada muestra a ser evaluada.
- Las herramientas vistas pueden ejecutarse en entornos HPC.