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

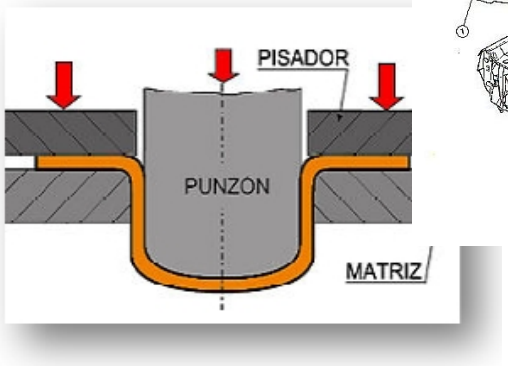
Varios ámbitos de aplicación:

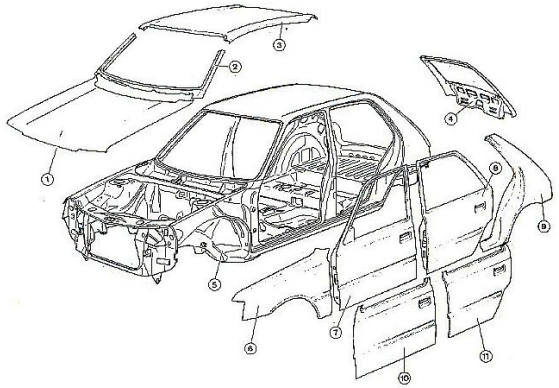
- Calidad (*Determinar el reemplazo de cilindros según defectos)*

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

Caso de ejemplo

SIDERURGIA => El caso del “cerrojo”:

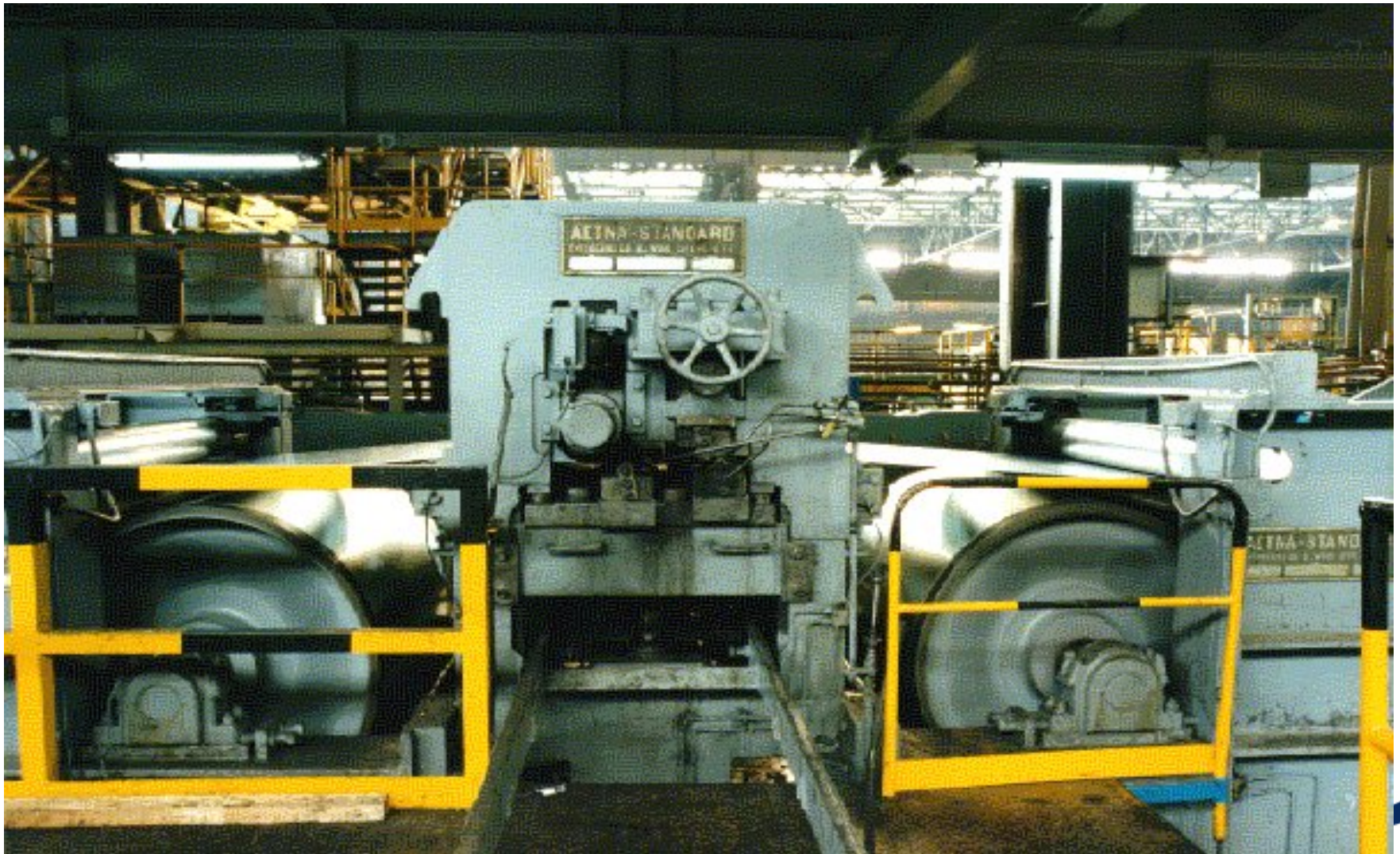




-Figura 18: Esquema embutición de doble efecto-

Caso de ejemplo

SIDERURGIA => El caso del “cerrojo”:



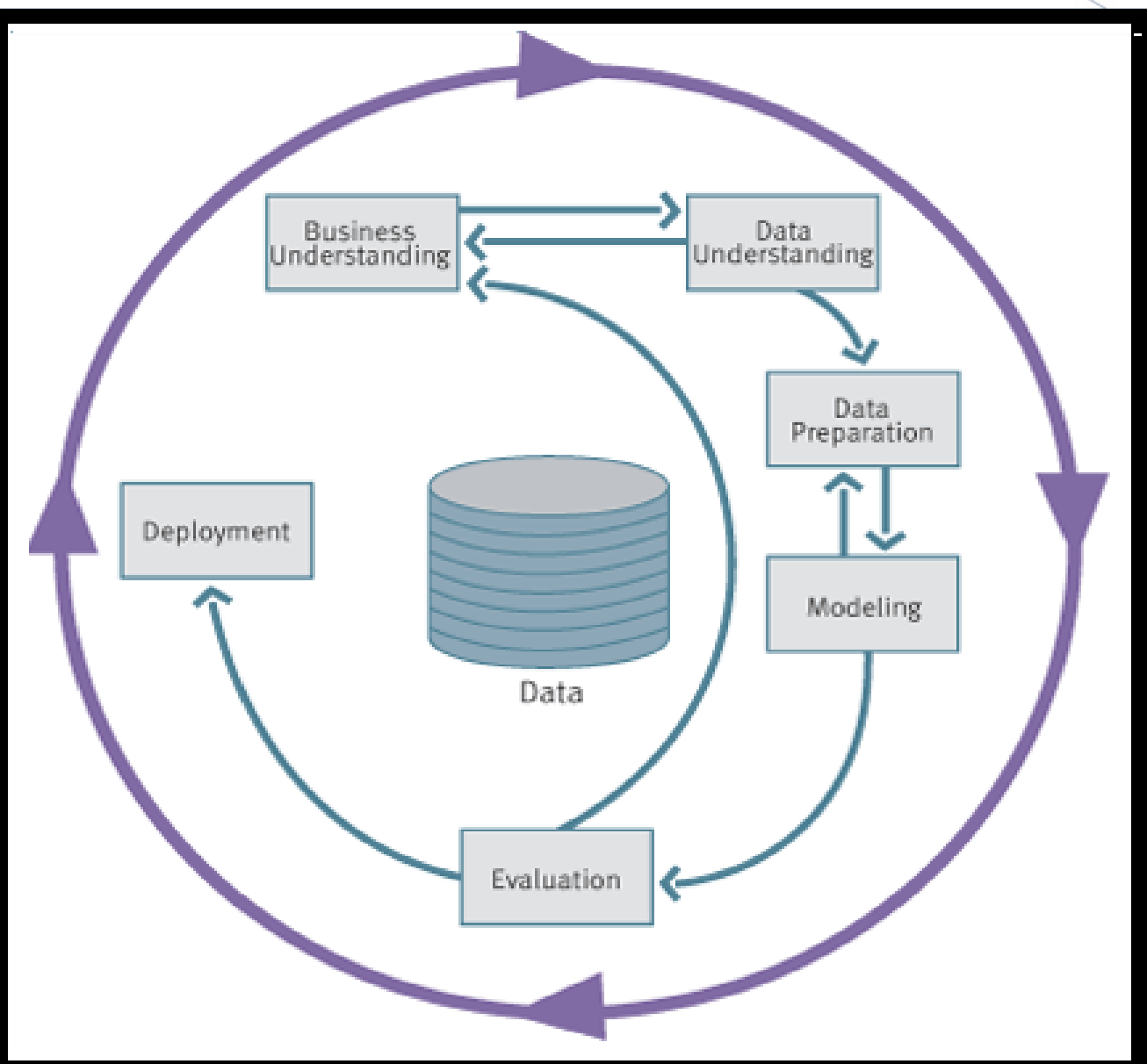
Metodología

CRISP/DM
Varios ámbitos de aplicación =>

Varias metodologías:

- CRISP/DM,
- Fayads,
- SEMMA,

etc.

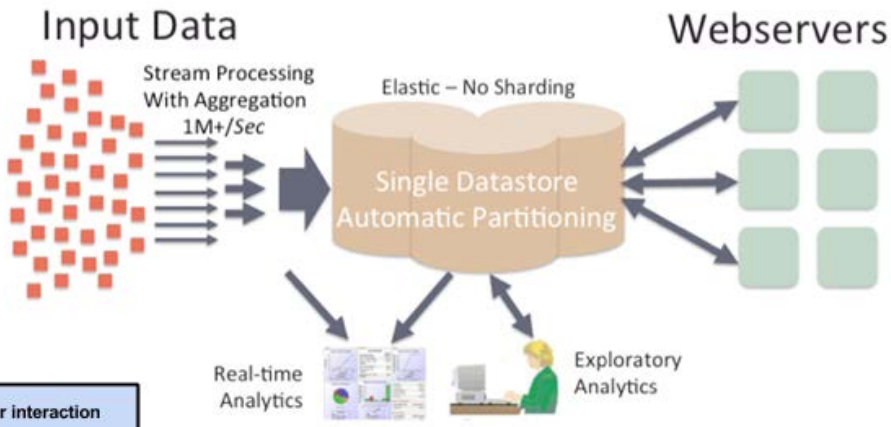


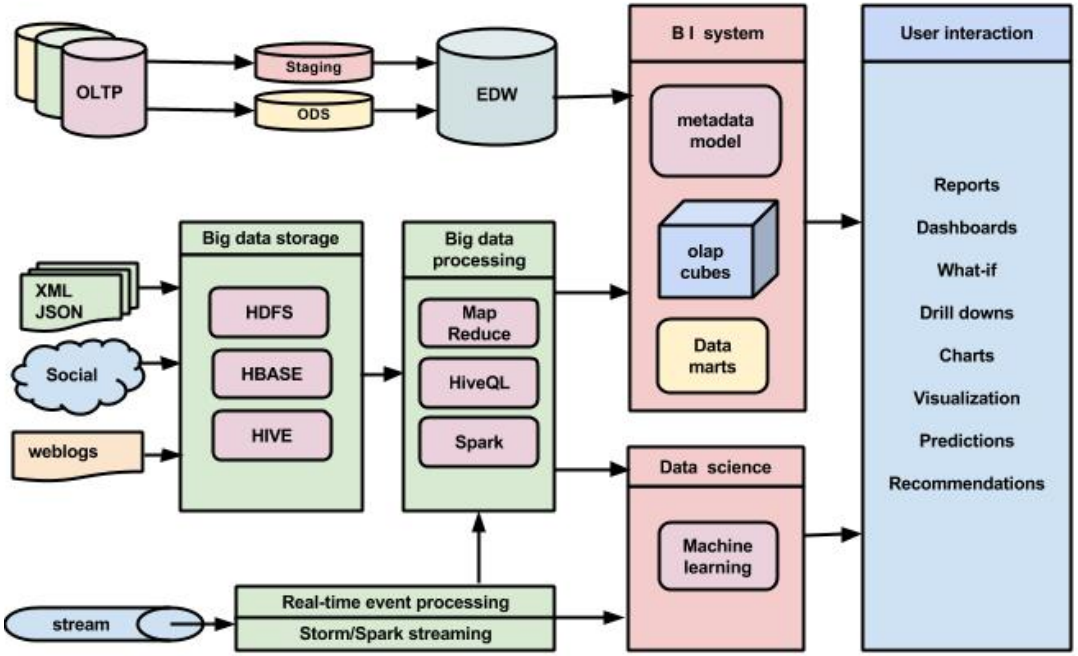
Metodología

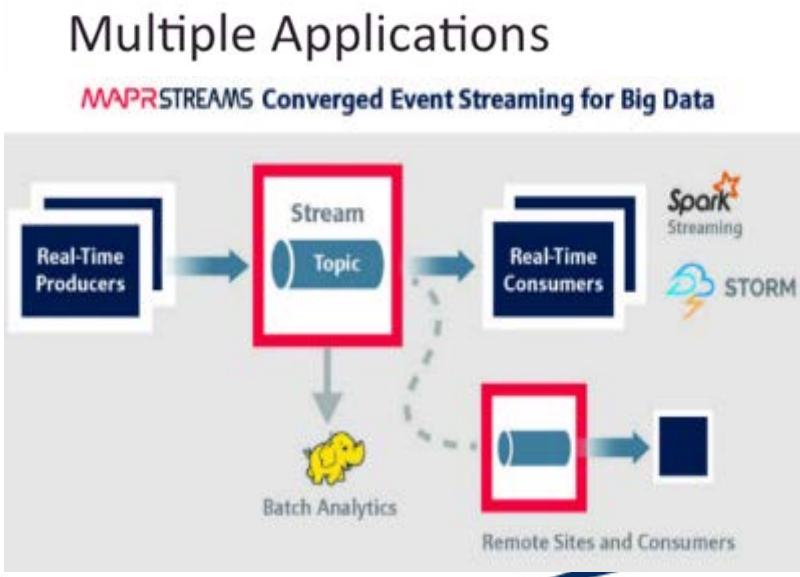
CRISP/DM / Data Preparation

Incluye:

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







Metodología

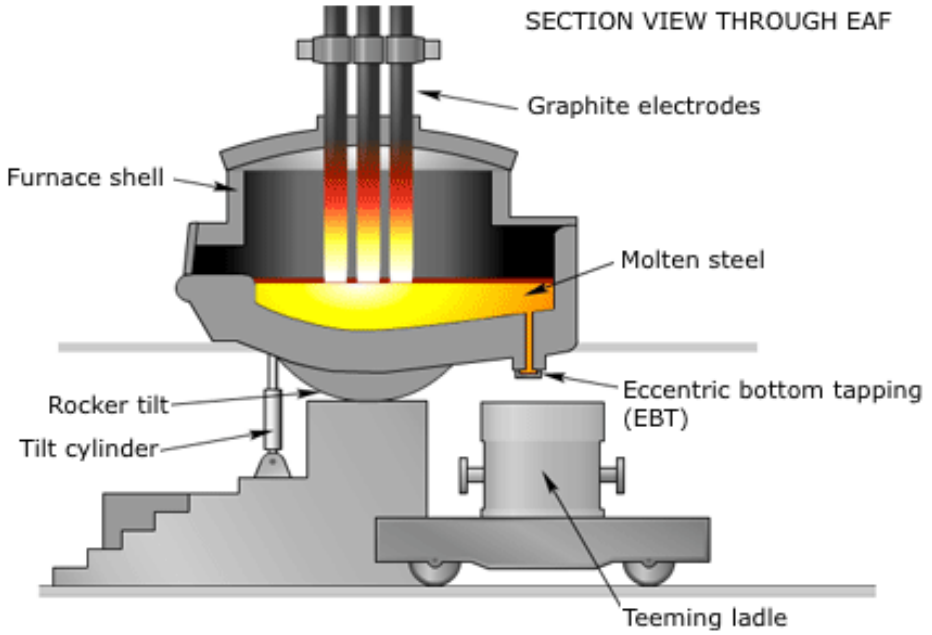
CRISP/DM / Data Preparation : EDA

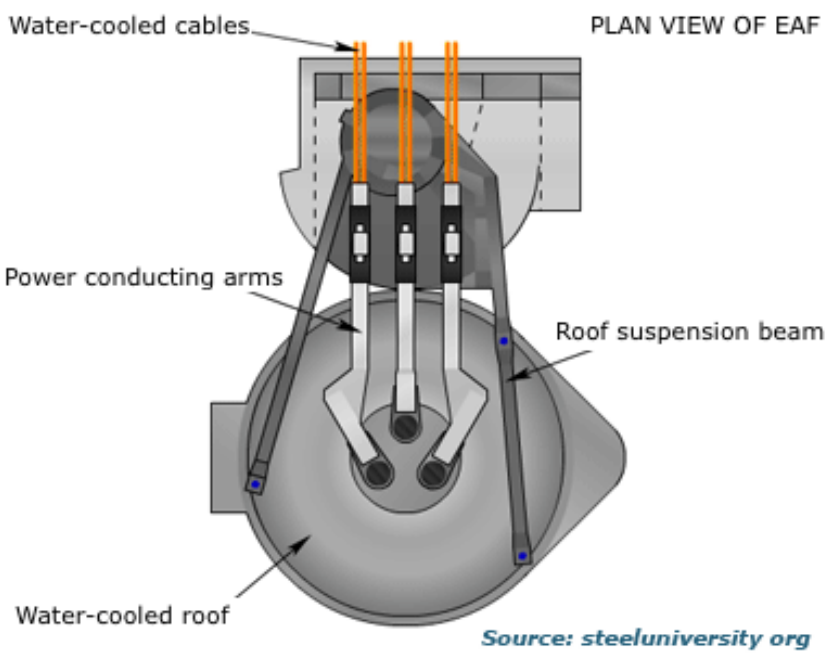
Metodología

CRISP/DM / Data Preparation : EDA

CRISP/DM / Data Preparation : EDA

Section and Plan View of Electric Arc Furnace



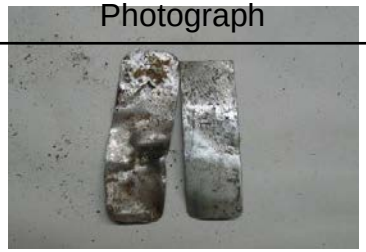


Metodología

CRISP/DM / Data Preparation : EDA

Metodología

CRISP/DM / Data Preparation : EDA

Scrap Type	Measured Composition	Photograph
English: ***4A Bundles*** Eurofer: ***E653*** Spanish: ***Paquete 4A***	***•C: 0.072 %*** ***•N: 24 ppm***	
English: ***4C Bundles*** Eurofer: ***E6*** Spanish: ***Paquete 4C***	***•C: 0.077 %*** ***•N: 35 ppm***	
English: ***Forging Flash*** Eurofer: ***E206*** Spanish: ***Estampación Forja***	***•C: 0.390 %*** ***•N: 135 ppm***	
English: ***7B (Machine Shop Turnings)*** Eurofer: ***E5M*** Spanish: ***Viruta Interna***	***•C: 0.293 %*** ***•N: 123 ppm***	

Metodología

CRISP/DM / Data Preparation : EDA

Scrap Type	Measured Composition	Photograph
English: ***OA / Structural*** Eurofer: ***E3*** Spanish: ***OA Estructural***	***•C: 0.346 %*** ***•N: 40 ppm***	
English: ***Shipbreaking scrap / Structural*** Eurofer: ***E3C*** Spanish: ***Estructural***	***•C: 0.095 %*** ***•N: 70 ppm***	
English: ***Scrap / Skull*** Eurofer: ***Scrap / Skull*** Spanish: ***Escarpa Recuperación Interna***	***•C: 0.289 %*** ***•N: 109 ppm***	
English: ***Pig Iron Ingot*** Eurofer: Spanish: ***Lingotillo***	***•C: 4.13 %*** ***•N: 50 ppm***	

Metodología

CRISP/DM / Data Preparation : EDA

E1.- C=0.5%

E3.- C=0.4%

E40.- C=0.3%

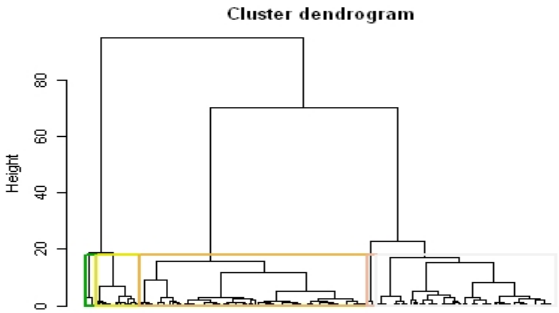
E2.- C=0.2%

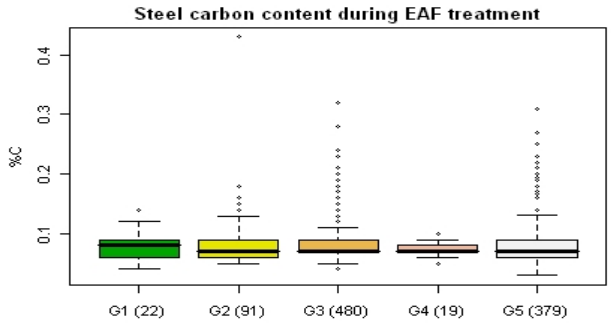
E6.- C=0.1%

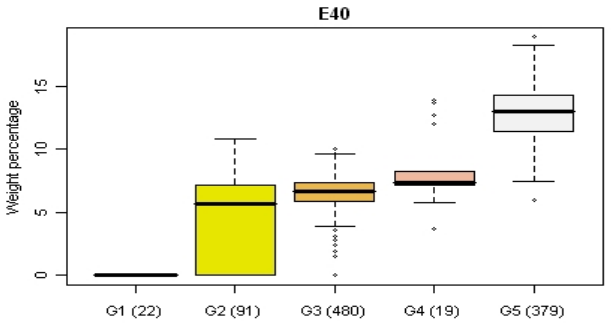
Beach Iron.- C=4.5%

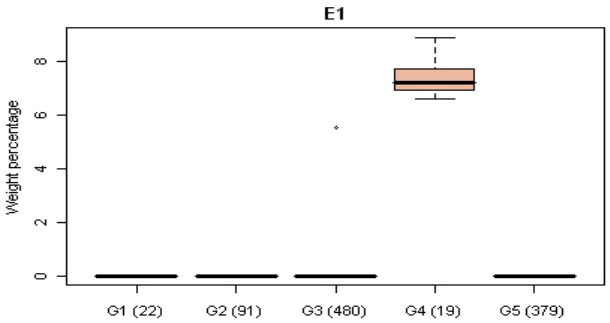
Metodología

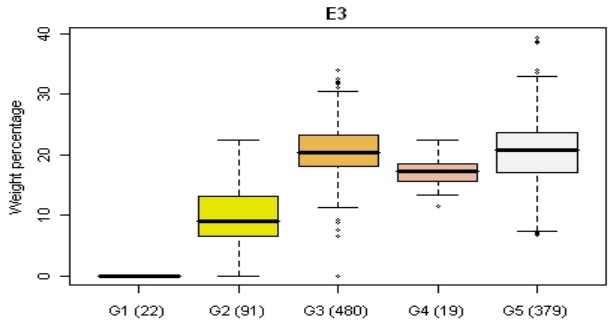
CRISP/DM / Data Preparation : EDA

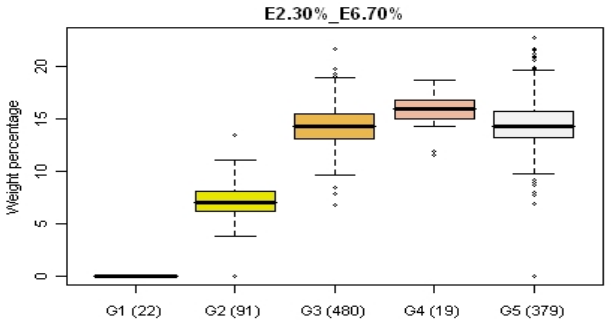


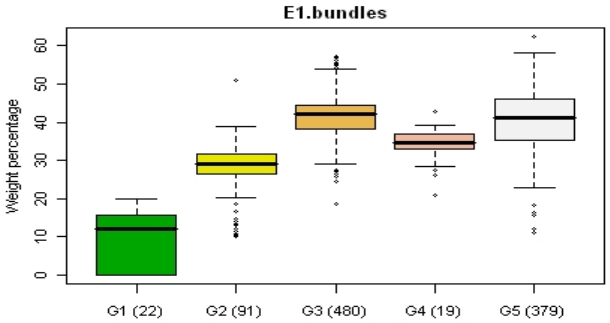


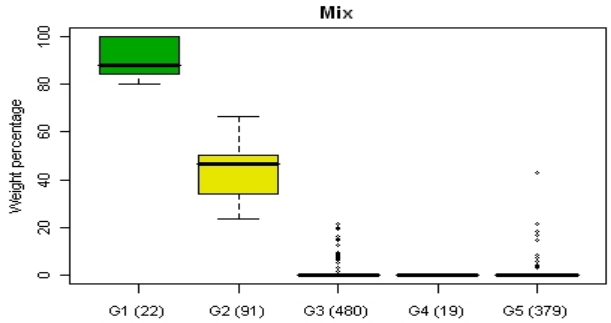


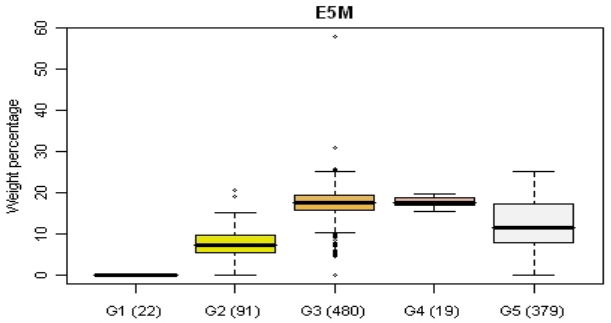




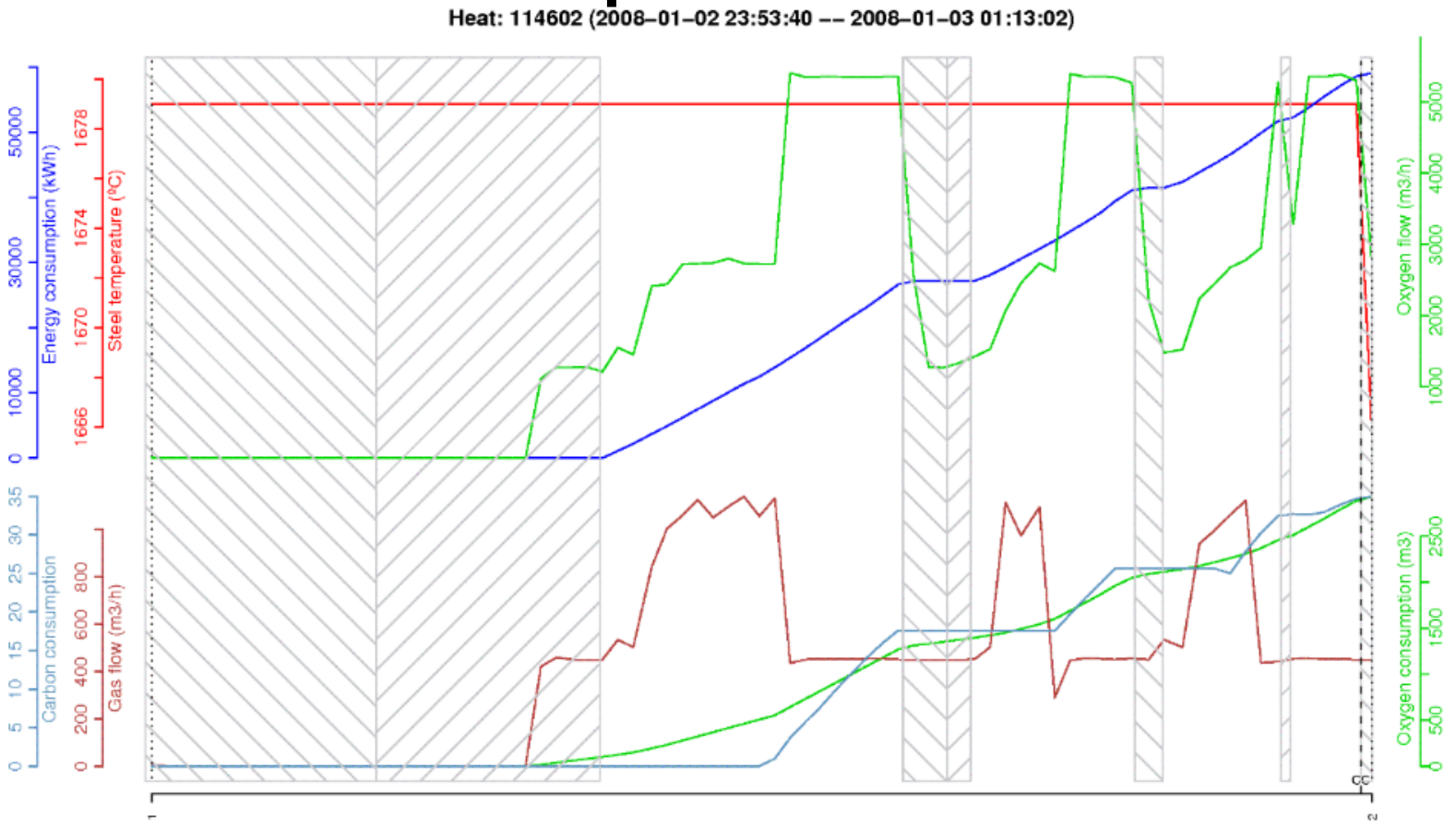








CRISP/DM / Data Preparation : EDA



Point	Time	min.	min.(%)	Temp.	kWh	kWh(%)
1	23:53:40	0	0	1679	0	0
2	01:13:02	79.37	100	1666	59100	100

CC (laf01)

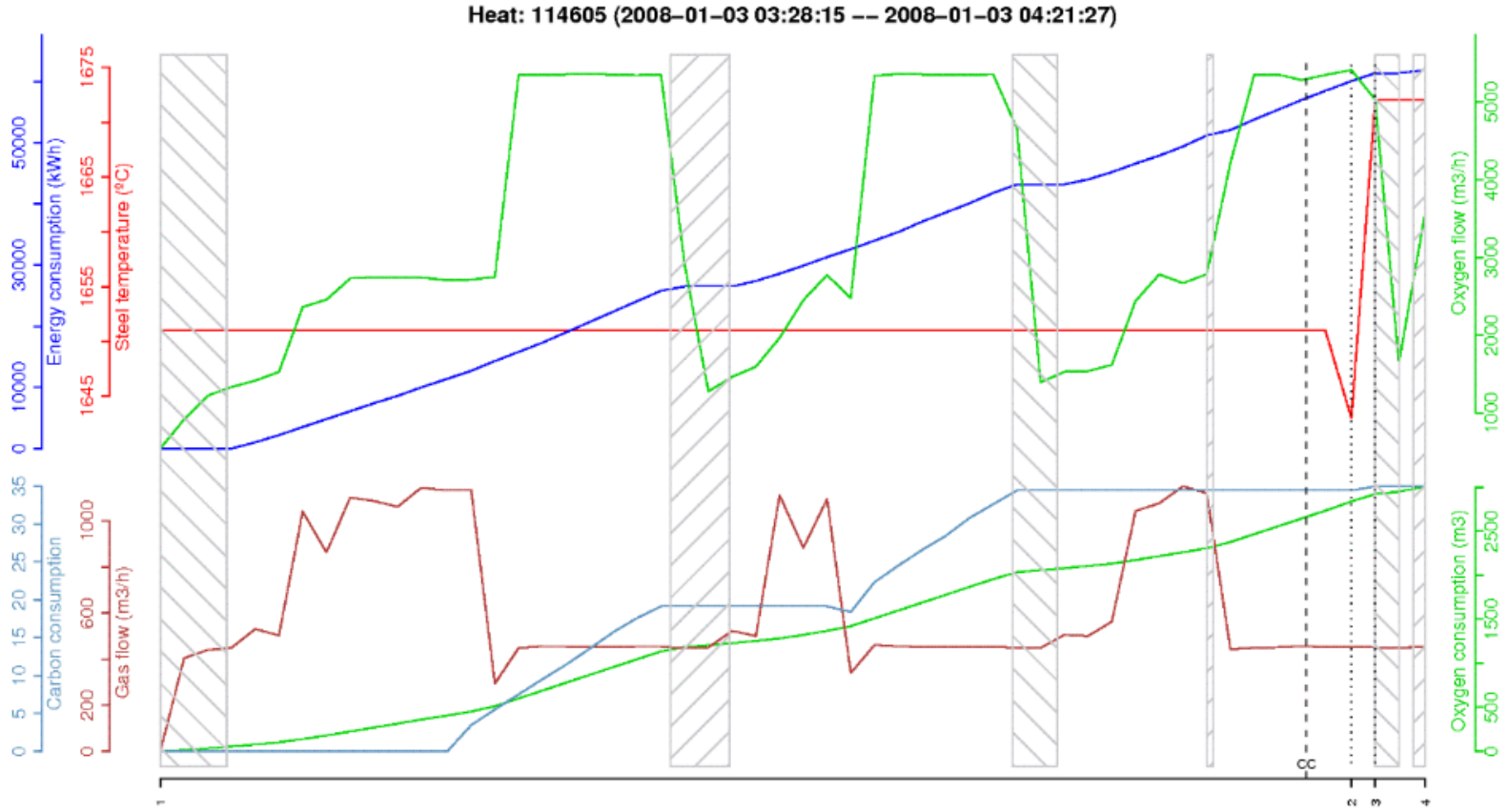
Time	C	P	Al
01:12:21	0.06	0.009	0.68
01:56:57	0.063	0.009	0.001
02:21:06	0.069	0.008	0.002
02:57:54	0.079	0.008	0.001

Paradas (eaf19)

Start	End	Duration	Description
23:53:16	00:08:16	0h 15' 0	GUNITADO REVESTIMIENTO HORNO
00:08:16	00:22:47	0h 14' 31	ARCOS
00:42:28	00:45:22	0h 2' 54	CESTA CON ADICIÓN DE CAL
00:45:22	00:46:52	0h 1' 30	PISAR CHATARRA
00:57:35	00:59:23	0h 1' 48	CESTA SIN ADICIÓN DE CAL
01:07:06	01:07:39	0h 0' 33	LIMPIAR PUERTA
01:12:21	01:21:03	0h 8' 42	COLAR (COLAR + CERRAR BUZA + CARGAR)

Metodología

CRISP/DM / Data Preparation : EDA



Point	Time	min.	min.(%)	Temp.	kWh	kWh(%)
1	03:28:15	0	0	1651	0	0
2	04:18:26	50.18	94.33	1643	60100	97.09
3	04:19:26	51.18	96.21	1672	61400	99.19
4	04:21:27	53.2	100	1672	61900	100

CC (laf01)

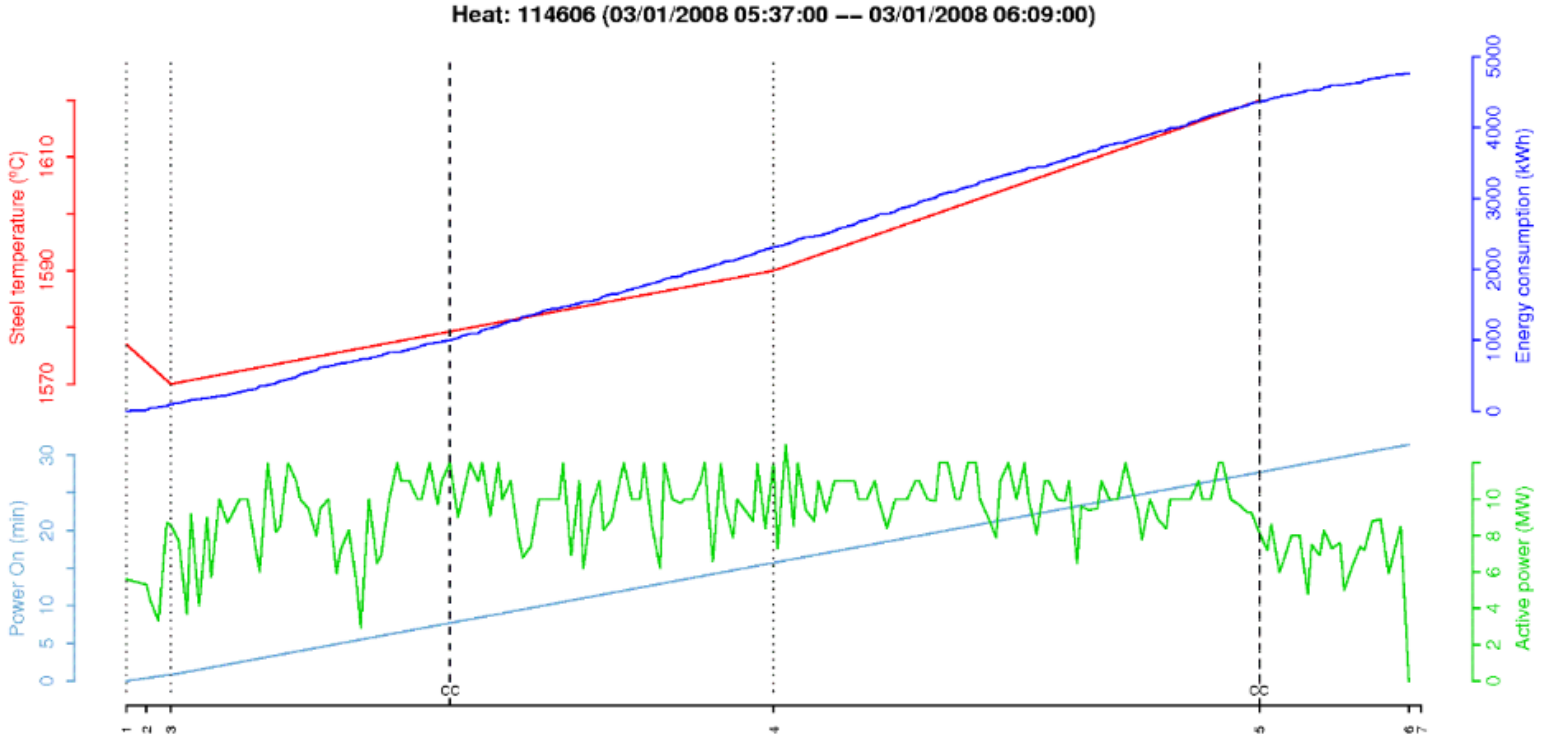
Time	C	P	Al
04:16:28	0.09	0.017	0.519
04:41:46	0.046	0.008	0.002
04:59:53	0.072	0.01	0.002
06:25:56	0.083	0.011	0.003

Paradas (eaf19)

Start	End	Duration	Description
03:23:31	03:31:01	0h 7' 30	COLAR (COLAR + CERRAR BUZA + CARGAR)
03:49:40	03:52:10	0h 2' 30	CESTA CON ADICIÓN DE CAL
04:04:05	04:05:57	0h 1' 52	CESTA SIN ADICIÓN DE CAL
04:12:16	04:12:36	0h 0' 20	LIMPIAR PUERTA
04:19:21	04:20:25	0h 1' 4	ABRIR BUZA
04:21:01	04:28:22	0h 7' 21	COLAR (COLAR + CERRAR BUZA + CARGAR)

Metodología

CRISP/DM / Data Preparation : EDA



Point	Time	min.	POn	ºC	kWh	table
1	05:37:00	0	–	1577	–	ste14
2	05:37:28	0.47	0	–	0	sucesos
3	05:38:09	1.15	0.9	1570	98	sucesos
4	05:52:59	15.98	15.7	1590	2310	sucesos
5	06:04:59	27.98	27.7	1620	4360	sucesos
6	06:08:40	31.67	31.35	–	4760	sucesos
7	06:09:00	32	0	1626	4760	ste14

CC (laf02)

Time	C	P	S	Al	Si
05:28:01	0.05	0.009	0.08	0.242	0.01
05:45:02	0.052	0.009	0.075	0.002	0.15
06:05:02	0.082	0.011	0.038	0.005	0.19
07:47:27	0.079	0.01	0.019	0.003	0.18
08:57:13	0.081	0.011	0.024	0.001	0.16

Anomalies (ste33)

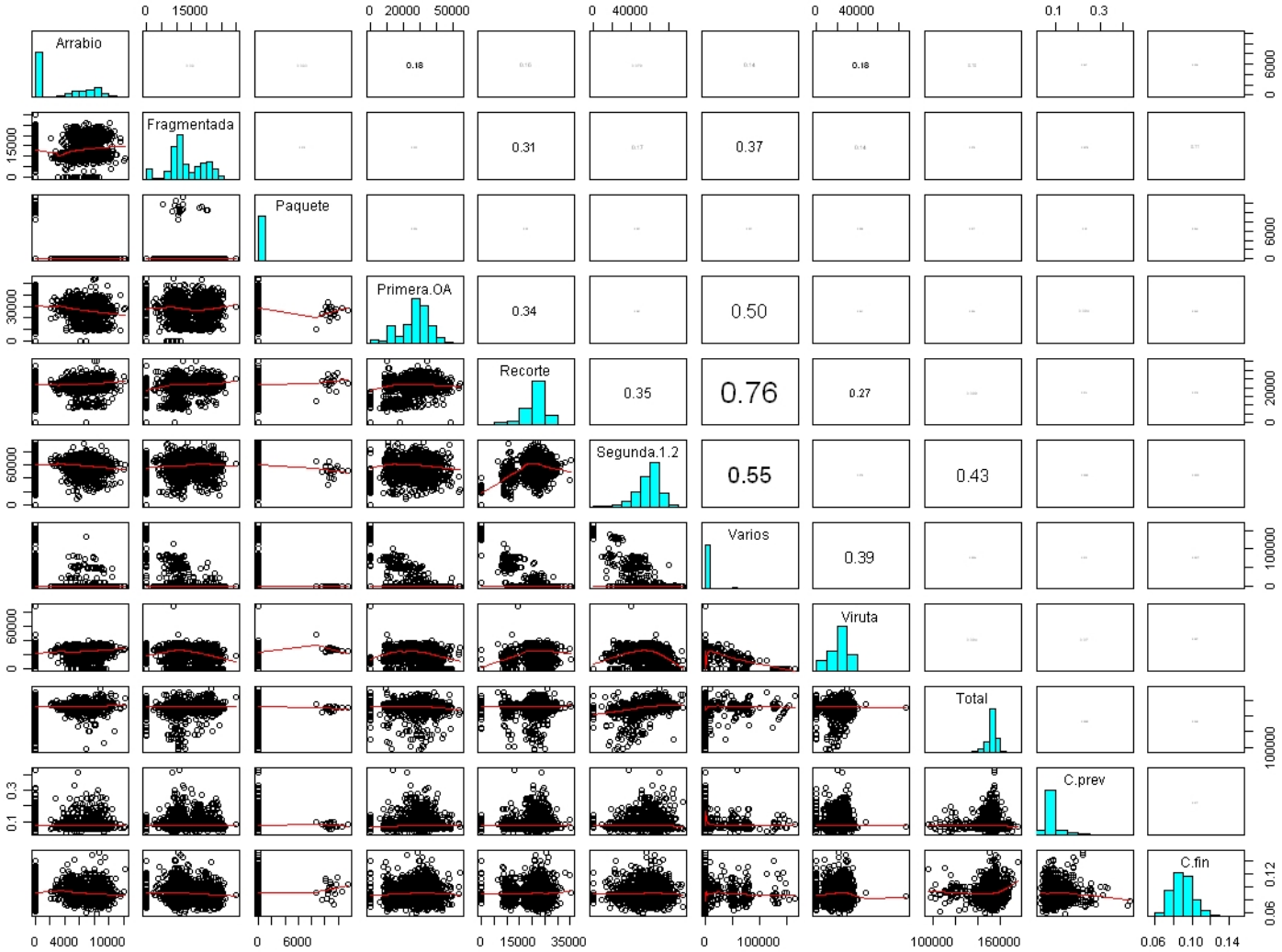
Location	Description

Additions (ste07.afino)

Lime	Carbon	Dolo-Lime	FeSi	SiMn	FeMn	FeAl	Spar	Graphite	FeV	FeNb	FeCr	FeC	FeTi
400	0	0	0	750	0	0	100	0	65	0	0	0	0

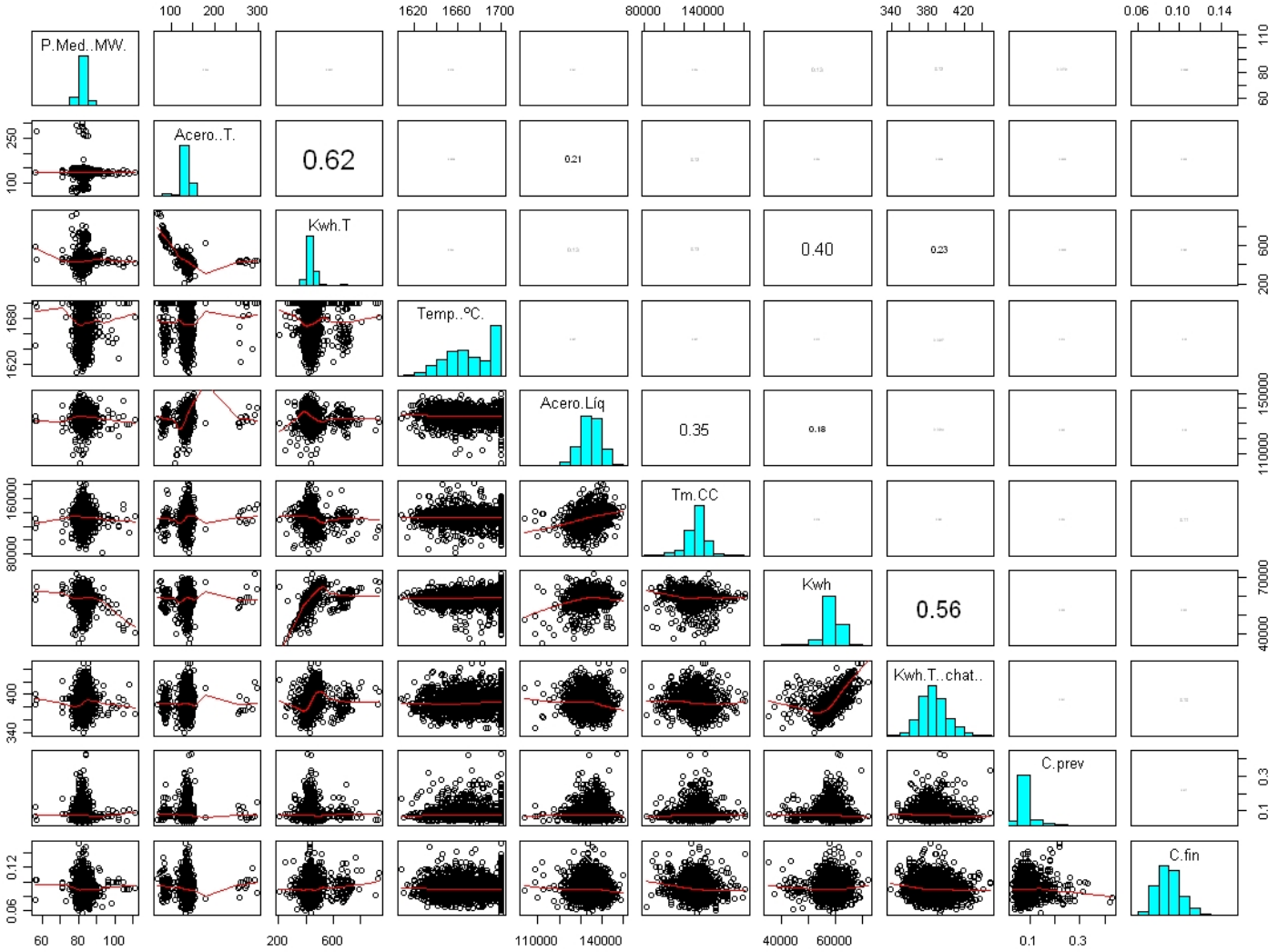
Metodología

CRISP/DM / Data Preparation : EDA



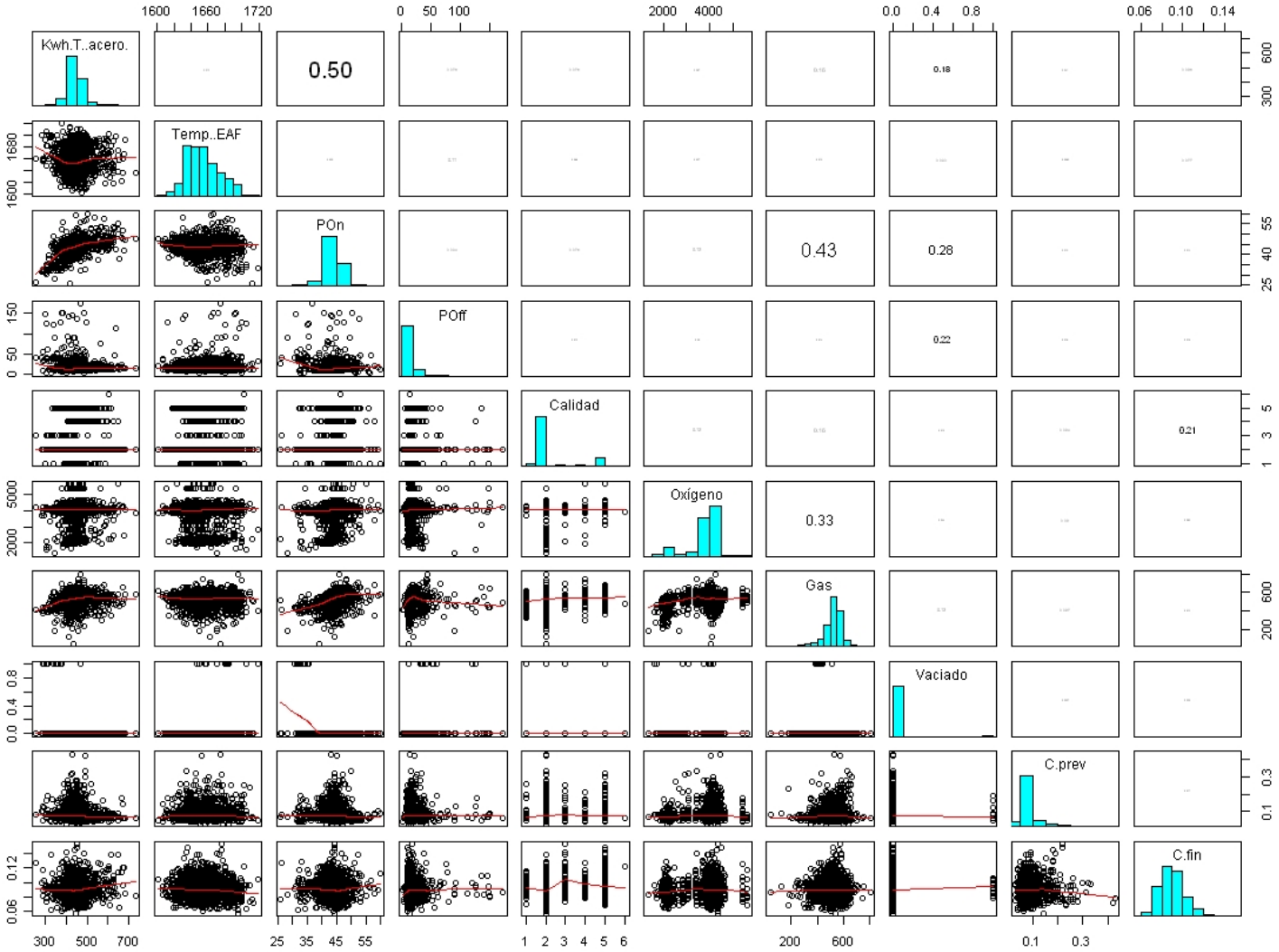
Metodología

CRISP/DM / Data Preparation : EDA



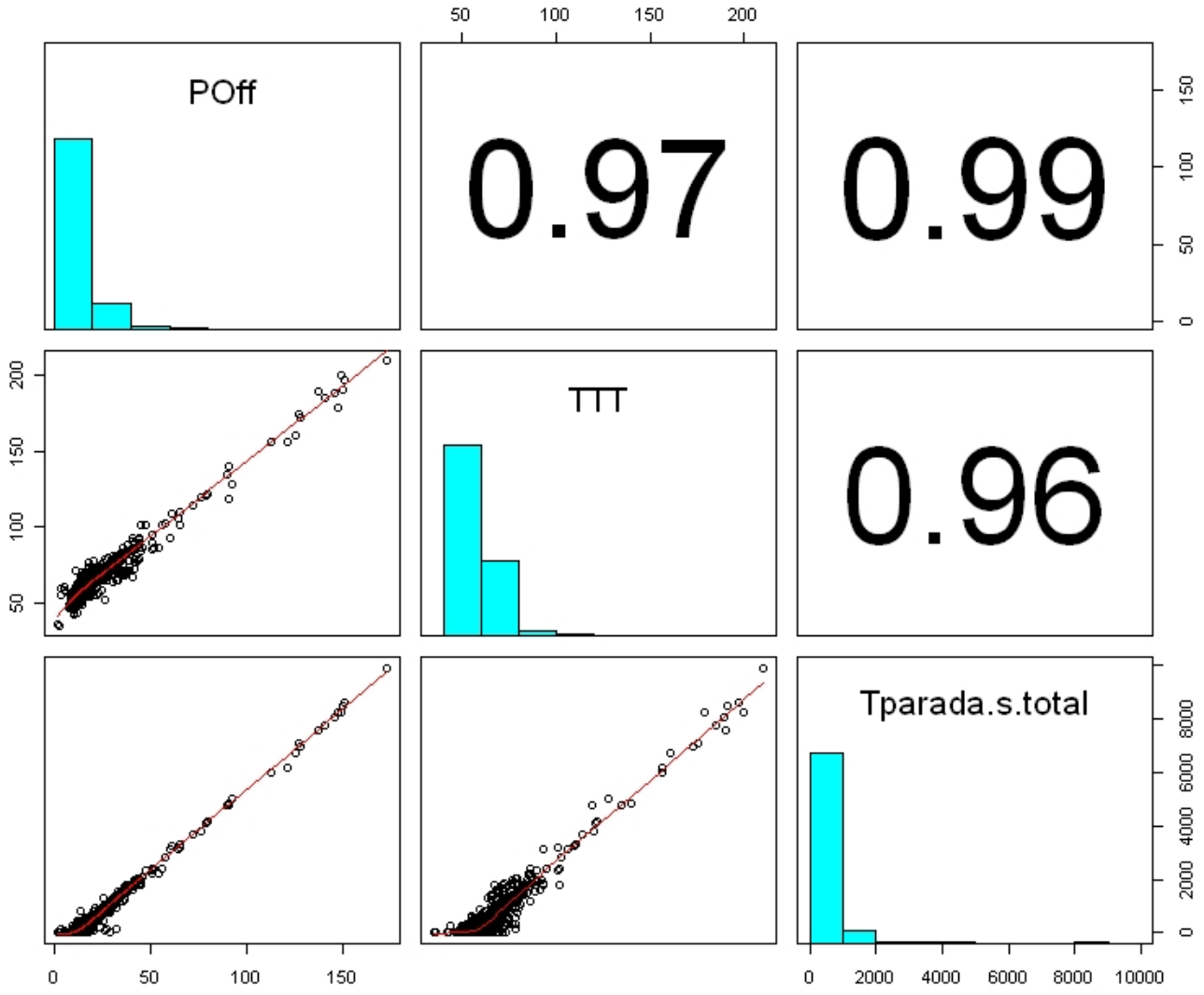
Metodología

CRISP/DM / Data Preparation : EDA

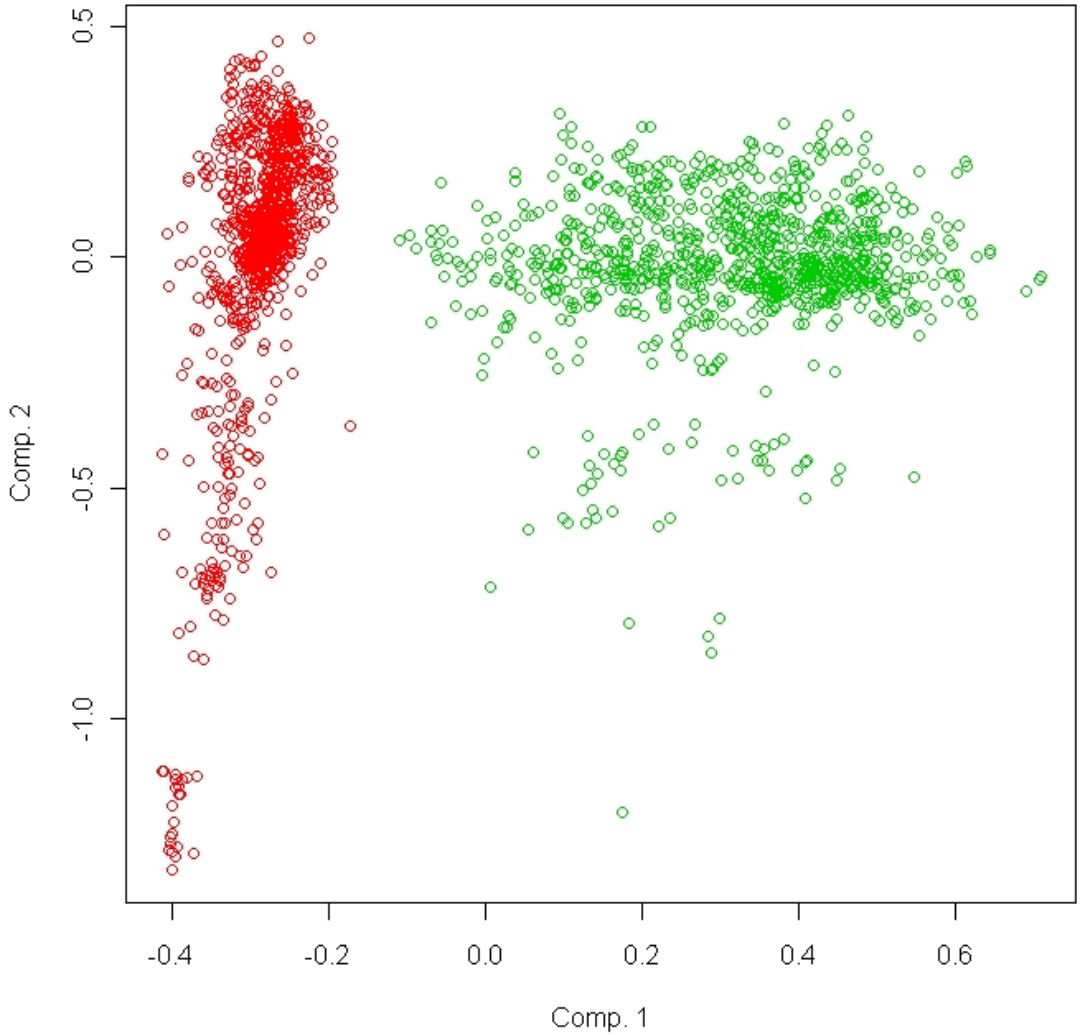


Metodología

CRISP/DM / Data Preparation : EDA



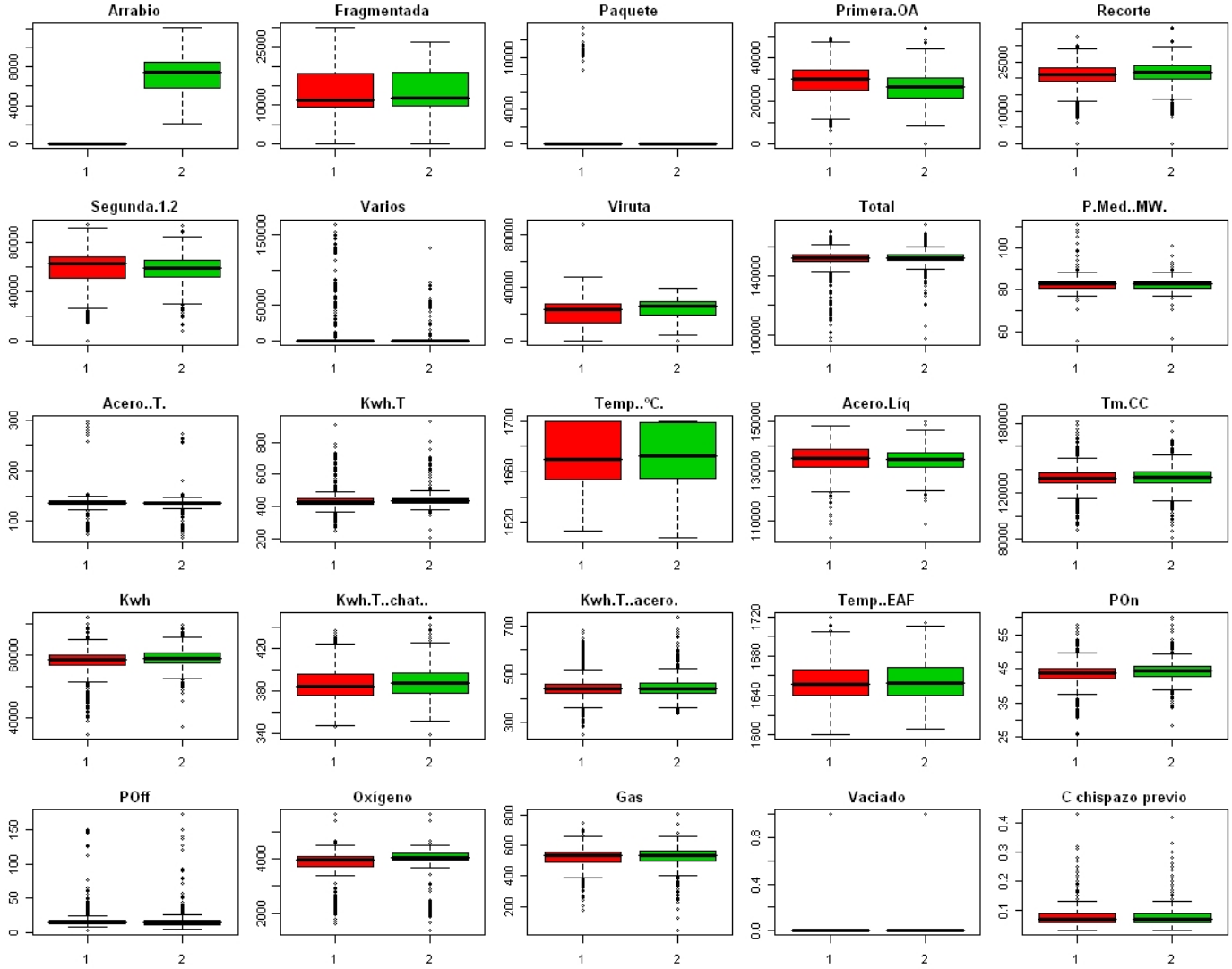
CRISP/DM / Data Preparation : EDA



La presencia de arrabio en la colada segmenta su comportamiento como se ve en el siguiente análisis.

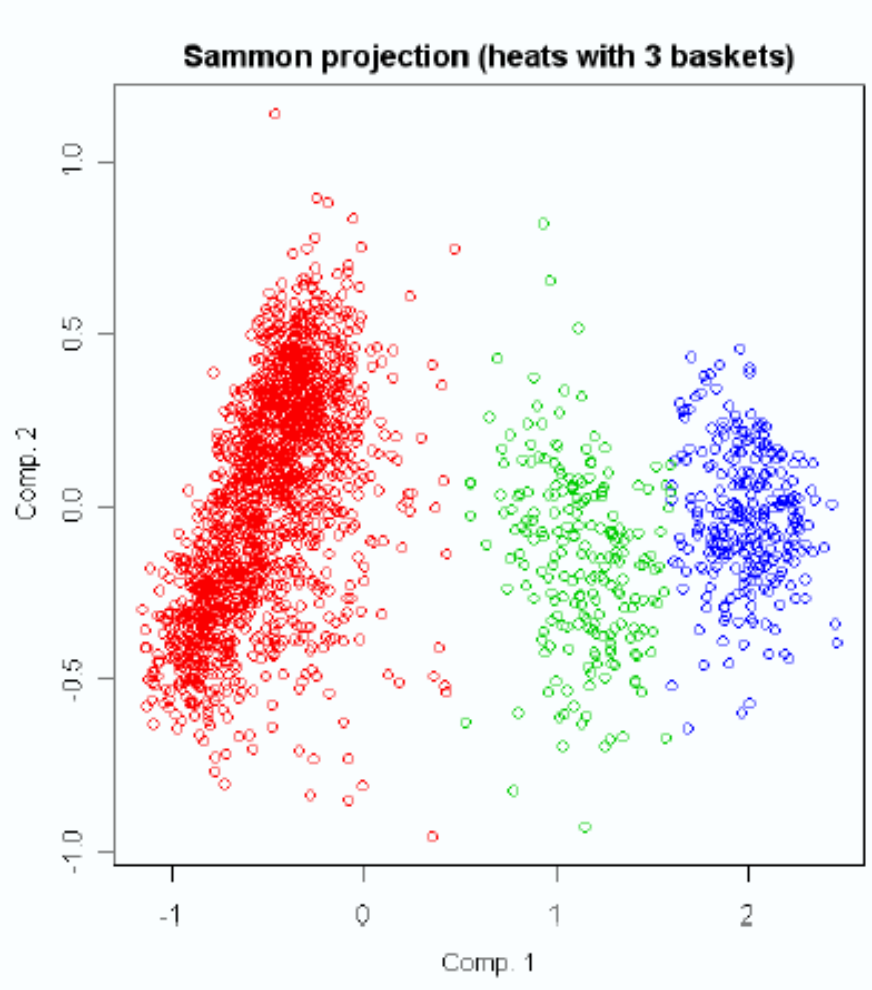
Metodología

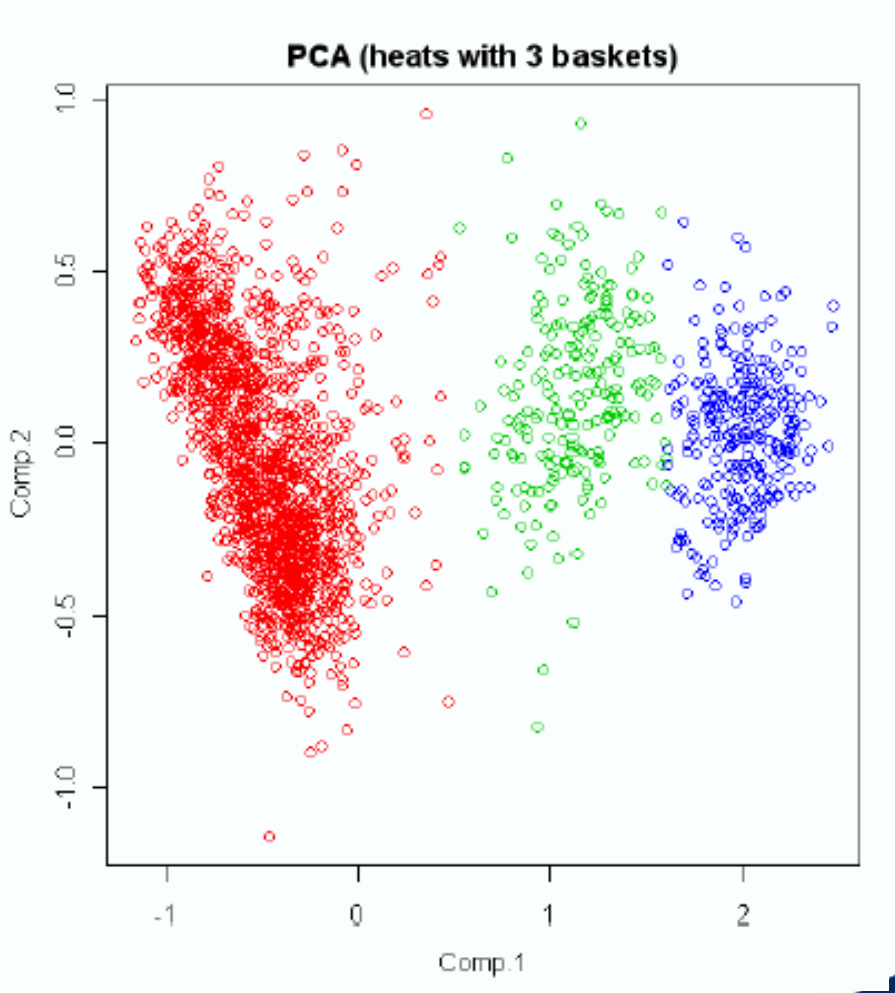
CRISP/DM / Data Preparation : EDA



Metodología

CRISP/DM / Data Preparation : EDA



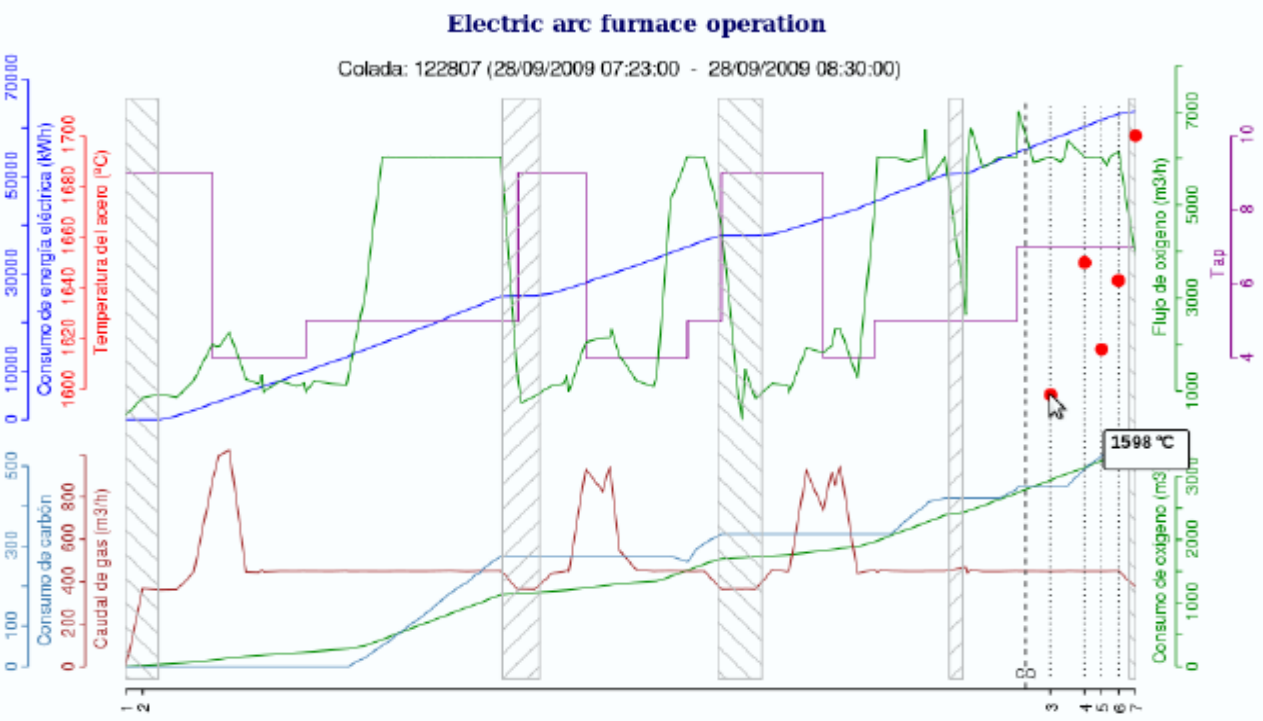


Metodología

CRISP/DM / Data Preparation : Embed modelling

Plot EAF and LF treatment

Predict S content in LF (first sample) %



Punto	Hora	min.	POn	Temp.	kWh	tabla
1	07:21:57	0	0	1690	0	sucesos
2	07:23:00	1.05	-	-	-	ste04
3	08:16:29	54.53	45.42	1598	57600	sucesos
4	08:18:29	56.53	47.42	1650	60300	sucesos
5	08:19:29	57.53	48.42	1616	61600	sucesos
6	08:20:29	58.53	49.42	1643	62900	sucesos
7	08:21:29	59.53	49.92	1700	63500	sucesos
-	08:30:00	-	51.38	1690	63500	ste04

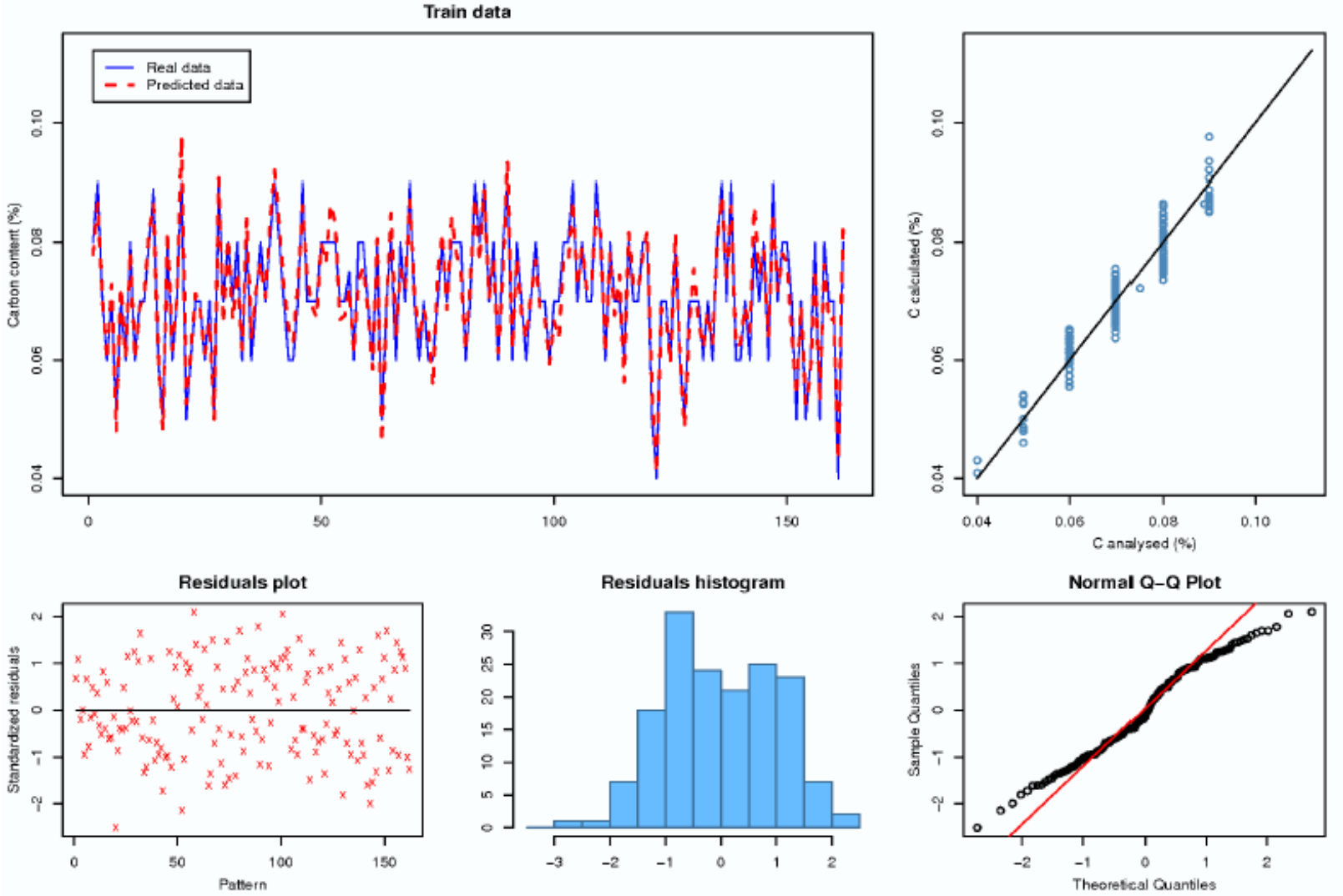
CQ (ler02)

	Hora	C	P	Al
HF	08:15:00	0.09	0.013	0.504
	08:37:00	0.031	0.019	0.002
HA	09:00:00	0.083	0.022	0.004
CC	10:09:00	0.111	0.023	0.004

Paradas (sel19)

Inicio	Fin	Duración	Descripción
07:17:30	07:23:56	0h 6' 26	PREPARACION HORNO
07:44:15	07:46:23	0h 2' 8	CESTA CON ADICI
07:56:56	07:59:32	0h 2' 36	CESTA CON ADICI
08:10:29	08:11:19	0h 0' 50	LIMPIAR PUERTA

CRISP/DM / Data Preparation : Embed modelling



Metodología

CRISP/DM / Data Preparation : EDA

```
```{r setup, include=FALSE, echo=FALSE, message=FALSE}
# devtools::install_github("hafen/trelliscopejs")
# devtools::install_github("vqv/ggbiplot")
# library(trelliscopejs)library(ggbiplot)
#
cargar=function (x) {
      if (! require(x,character.only = TRUE)) {
        install.packages(x,repos="http://cran.rstudio.com/")
        require(x,character.only = TRUE)
      }
}
#
pkgs=c('dplyr','ggplot2','plyr','zoo','stringr','utils','parallel','xt
able','readr','plotly','htmlwidgets',
          'tibble','trelliscopejs','rbokeh','dplyr','ggiraph',
‘xtable','SimilarityMeasures','DBI',
          'reshape2','scales', 'formattable',
'svglite','lubridate','grid','gridExtra','cowplot’)
lapply(pkgs, cargar)
#
```
```

CRISP/DM / Data Preparation : EDA

```
```{r lectura,warning=FALSE}
#
files = list.files(path='~/git/Salas/', pattern = '^[0-9].*.csv$')
nfiles= length(files)
d = list()
for ( i in files) {
   cat(paste("Procesando ",i,".\n",sep=""))
   d[[i]]  = read.csv2(file=i,sep=",",dec=".",
            header=TRUE, stringsAsFactors = FALSE)
}
```
```

Metodología

CRISP/DM / Data Preparation : EDA

```
setwd('~/git/Salas')
if (file.exists("MatDat.RData")) {
   load(file="MatDat.RData")
}
if ( ! exists(“matdat”) {
   for (i in ls(d)) {
      if ( ! exists("dd")) {dd=unique(as.Date(d[[i]][,1]))  }
      dd = cbind(dd,unique(as.Date(d[[i]][,1])))
   }
   dd = as.Date(unique(sort(dd)))
   matdat = data.frame(fecha=dd)
}
if (ncol(matdat) < (2*length(ls(d))+1)) {
  for (i in ls(d)) {
     matdat[,(ncol(matdat)+1):(ncol(matdat)+2)] =
        ldply(dd,function(x,d){return(range(which(as.Date(d)==x)))},d[
[i]][,1])
     cols = paste(i,c("-s","-e"),sep="")
     colnames(matdat)[(ncol(matdat)-1):ncol(matdat)] = cols
  }
  save(matdat,dd,file="MatDat.RData")
}
```

Metodología

CRISP/DM / Data Preparation : EDA

```
procesa = function(x,d,m){
    cols = ls(d)
    rowj = which(m[,1]==x)
    i= 1  ; ini  = cols[i]
    if ( ! is.infinite(as.numeric(m[rowj,paste(ini,"-s",sep="")]))) {
        d0  = d[[ini]][m[rowj,paste(ini,"-s",sep="")]:m[rowj,
                paste(ini,"-e",sep="")],]
        colnames(d0) = c("X",paste(colnames(d0)[-1],"-",ini,sep=""))
    }
    while (length(cols) > 1 && i < length(cols)) {
        i = i + 1
        ini = cols[i]
        if ( ! is.infinite(as.numeric(m[rowj,paste(ini,"-s",sep="")]))) {
           d1 = d[[ini]][m[rowj,paste(ini,"-s",sep="")]:m[rowj,
                   paste(ini,"-e",sep="")],]
           colnames(d1) = c("X",paste(colnames(d1)[-1],"-",ini,sep=""))
           d0 = merge(d0,d1,by="X")
        }
    }
    return(d0)
}
ldat = lapply(dd,procesa,d,matdat)
names(ldat)=dd
```

Metodología

CRISP/DM / Data Preparation : EDA

```
pgplt = function(dat,fechas,ID,vsen,ylb,numt,spt) {
    ddt = dat[[as.character(fechas[ID])]]
    colnames(ddt)[1]="Date"
    mdf = melt(ddt, id.vars="Date", value.name="value")
    mdf[,"Sensor"]= apply(mdf[,c("Date","variable")],1, function(x){return(
                  strsplit(x[2],"-")[[1]][1])})
    mdf[,"Sala"] = apply(mdf[,c("Date","variable")],1, function(x){return(
                  sub(".csv","", strsplit(x[2],"-")[[1]][3]))})
    mdf = mdf[mdf$Sensor == vsen,]
    mdf[,"Tiempo"]= substr(mdf[,"Date"],12,100)
    mdf[,"Date"] = as.numeric(as.POSIXct(mdf[,"Date"], format="%Y-%m-%d
                              %H:%M:%S"))
    lbs  = (trans_breaks(identity,identity,n=numt) (range(mdf[,"Date"])))
    tlbs = substr(as.POSIXct(as.numeric(lbs), origin="1970-01-01
    00:00:00"),9,16)
     p   = (ggplot(data=mdf, aes(x=Date, y=value, group= Sala, colour = Sala)) +
    geom_line() +  theme(axis.text.x =element_text(angle = 90,hjust = 1)) +
      scale_x_continuous(paste("Date: ",dd[ID], "->", weekdays(dd[ID]),sep=""),
                  breaks = lbs, labels = tlbs) +
              scale_y_continuous(eval(parse(text=expression(ylb))))+
              geom_point( size=spt, shape=21, fill="white"))
    return(p)
}
```

CRISP/DM / Data Preparation : EDA

```
vsen= c("CO2","Temperature","Humidity")
unts= c("CO[2] {ppm}","T {K}","RH {%}")
numt= 8
spt = 1.25
#
matd = data.frame(ID=1:length(dd),Date=dd,WeekDay=weekdays(dd),
                  Sensor=vsen[1], Unit=unts[1])
for (j in 2:length(vsen)) {
         matd = rbind(matd,data.frame(ID=1:length(dd),Date=dd,
                  WeekDay=weekdays(dd),Sensor=vsen[j], Unit=unts[j]))
}
#
fechas = matd[,"Date"]
matd[,"Sensor"] = as.character(matd[,"Sensor"])
matd[,"Unit"]   = as.character(matd[,"Unit"])
```

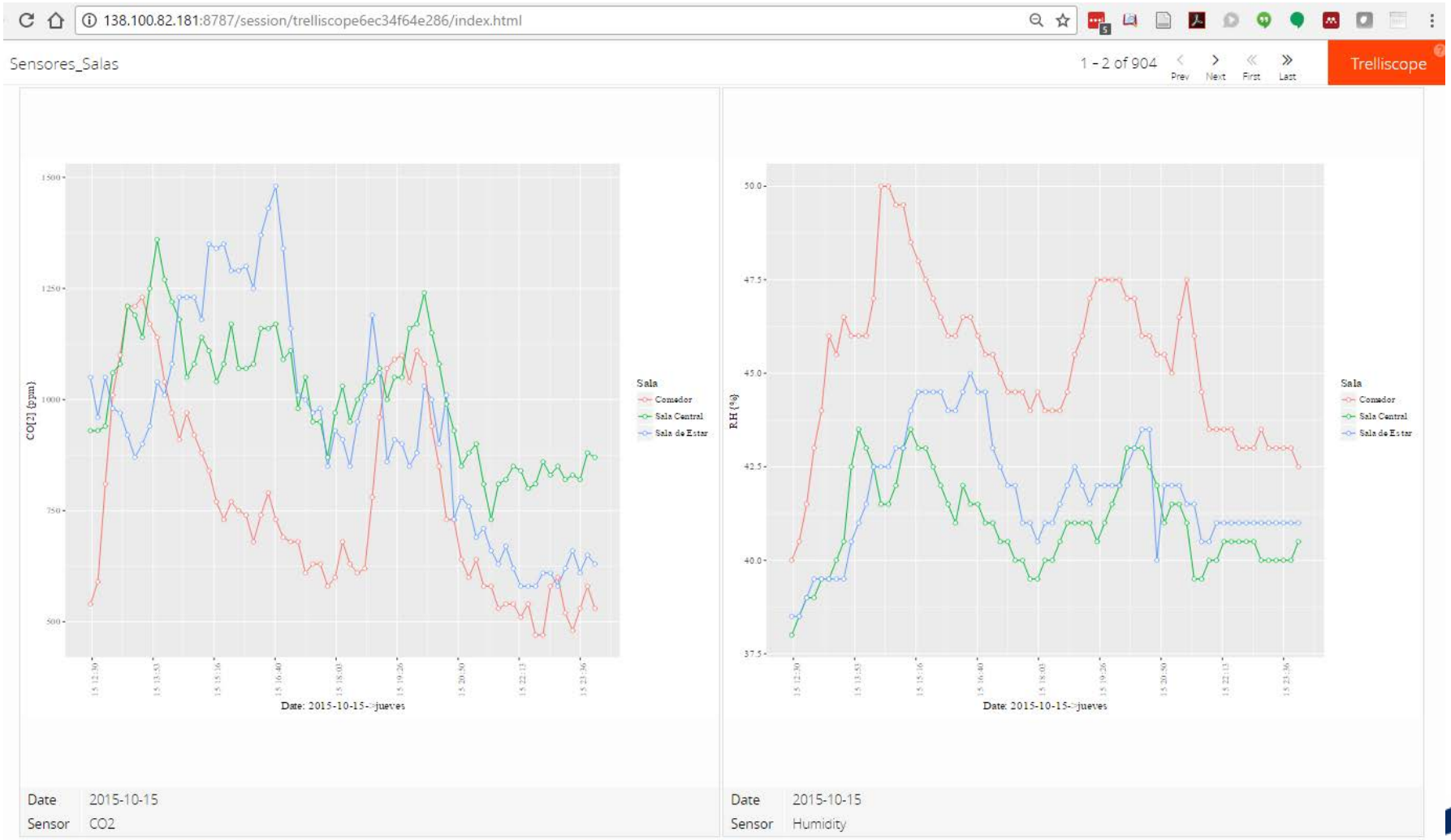
Metodología

CRISP/DM / Data Preparation : EDA

```
setwd('~/git/Salas')
for (ir in 1:nrow(matd)) {
   idd = matd[ir,"ID"]
   fec = matd[ir,"Date"]
   sns = as.character(matd[ir,"Sensor"])
   unt = as.character(matd[ir,"Unit"])
   fn  = paste("~/git/P01/imgs/",fec,"_",sns,".svg",sep="")
   if ( ! file.exists(fn)) {
     p = pgplt(ldat,fechas,idd,sns,unt,numt,spt)
#    htmlwidgets::saveWidget(ggiraph(code={print(p)}), fn)
     ggsave(file=fn,plot=p,width=10,height=8)
  }
  en  = paste("imgs/",fec,"_",sns,".svg",sep="")
  matd[ir,"panel"]=paste("http://138.100.82.181/~xxx/Salas_", en,sep="")
}
pg <- matd %>% mutate(img = img_panel(panel)) %>%
        trelliscope(name = "Sensores_Salas",nrow=3,ncol=3, width=800,
                    height=800,self_contained = FALSE,
                    state=list(labels=c("Date","Sensor")))
htmlwidgets::saveWidget(pg,"~/git/Salas/Ambiente.html")
pg
```

Metodología

CRISP/DM / Data Preparation : EDA



CRISP/DM / Data Preparation : EDA

```
#
extrae_hora = function(listax,hora,lugar) {
 desde = str_split(hora,"-")[[1]][1]
 hasta = str_split(hora,"-")[[1]][2]
 desde = paste(desde,":00",sep="")
 desde = paste(strftime(listax$X[1],"%Y-%m-%d")," ",desde,sep="")
 hasta = paste(strftime(listax$X[1],"%Y-%m-%d")," ",hasta,sep="")
 desde = as.POSIXct(desde)
 hasta = as.POSIXct(hasta)
 rd   = listax[(listax$X >= desde & listax$X <=hasta),]
 clss  = grep(lugar,colnames(rd))
 rd   = rd[,clss]
 clss  = colnames(rd)
 colnames(rd) = unlist(lapply(str_split(colnames(rd),"-
"),function(x){return(x[1])}))
 return(rd)
}
```

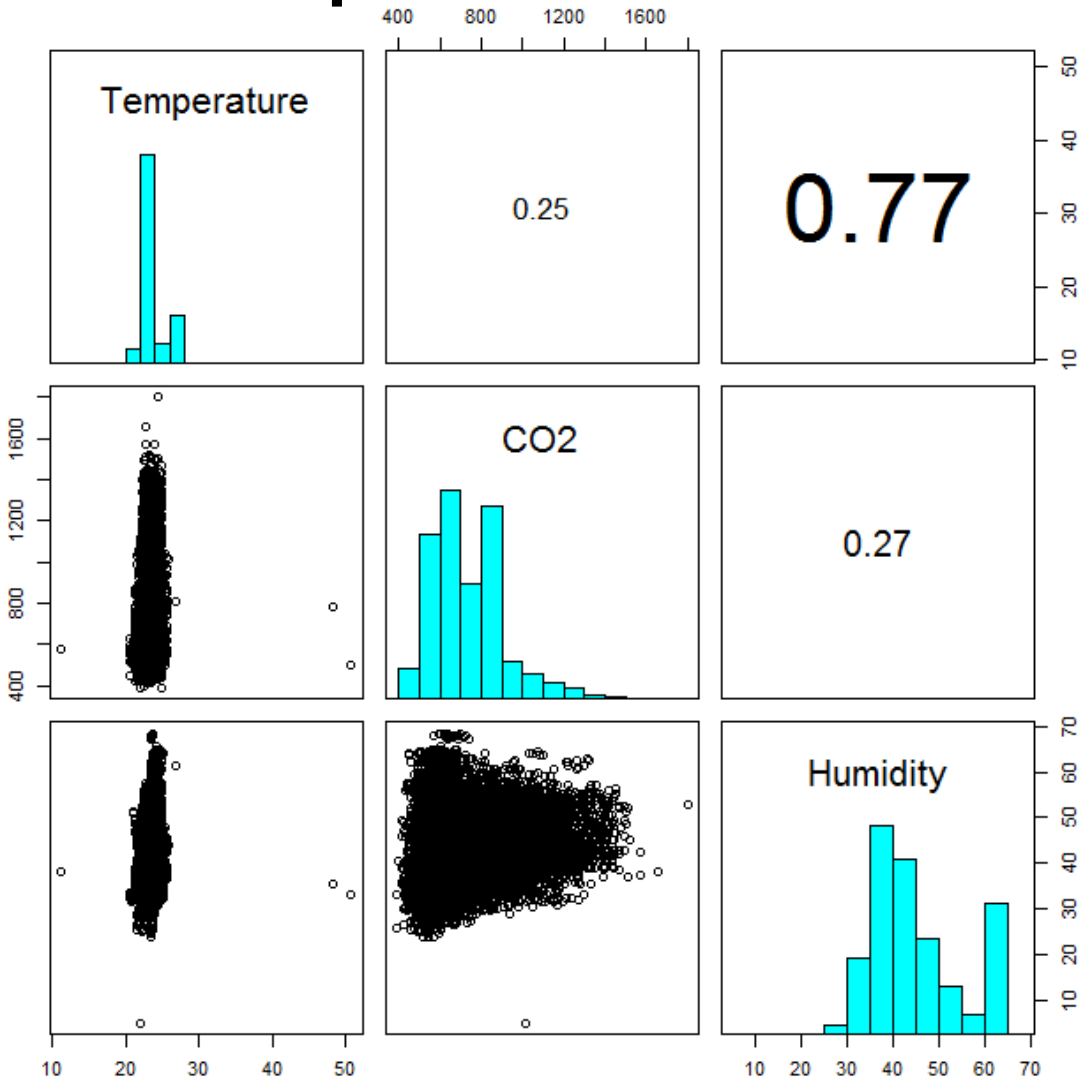
Metodología

CRISP/DM / Data Preparation : EDA

```
panel.hist <- function(x, ...){
   usr <- par("usr"); on.exit(par(usr))
   par(usr = c(usr[1:2], 0, 1.5) )
   h <- hist(x, plot = FALSE)
   breaks <- h$breaks; nB <- length(breaks)
   y <- h$counts; y <- y/max(y)
   rect(breaks[-nB], 0, breaks[-1], y, col="cyan", ...)
}
#
panel.cor <- function(x, y, digits=2, prefix="", cex.cor, ...){
   usr <- par("usr"); on.exit(par(usr))
   par(usr = c(0, 1, 0, 1))
   r <- abs(cor(x, y))
   txt <- format(c(r, 0.123456789), digits=digits)[1]
   txt <- paste(prefix, txt, sep="")
   if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
   text(0.5, 0.5, txt, cex = cex.cor * r)
}
#
pairs(ldd,upper.panel=panel.cor,diag.panel=panel.hist)
```

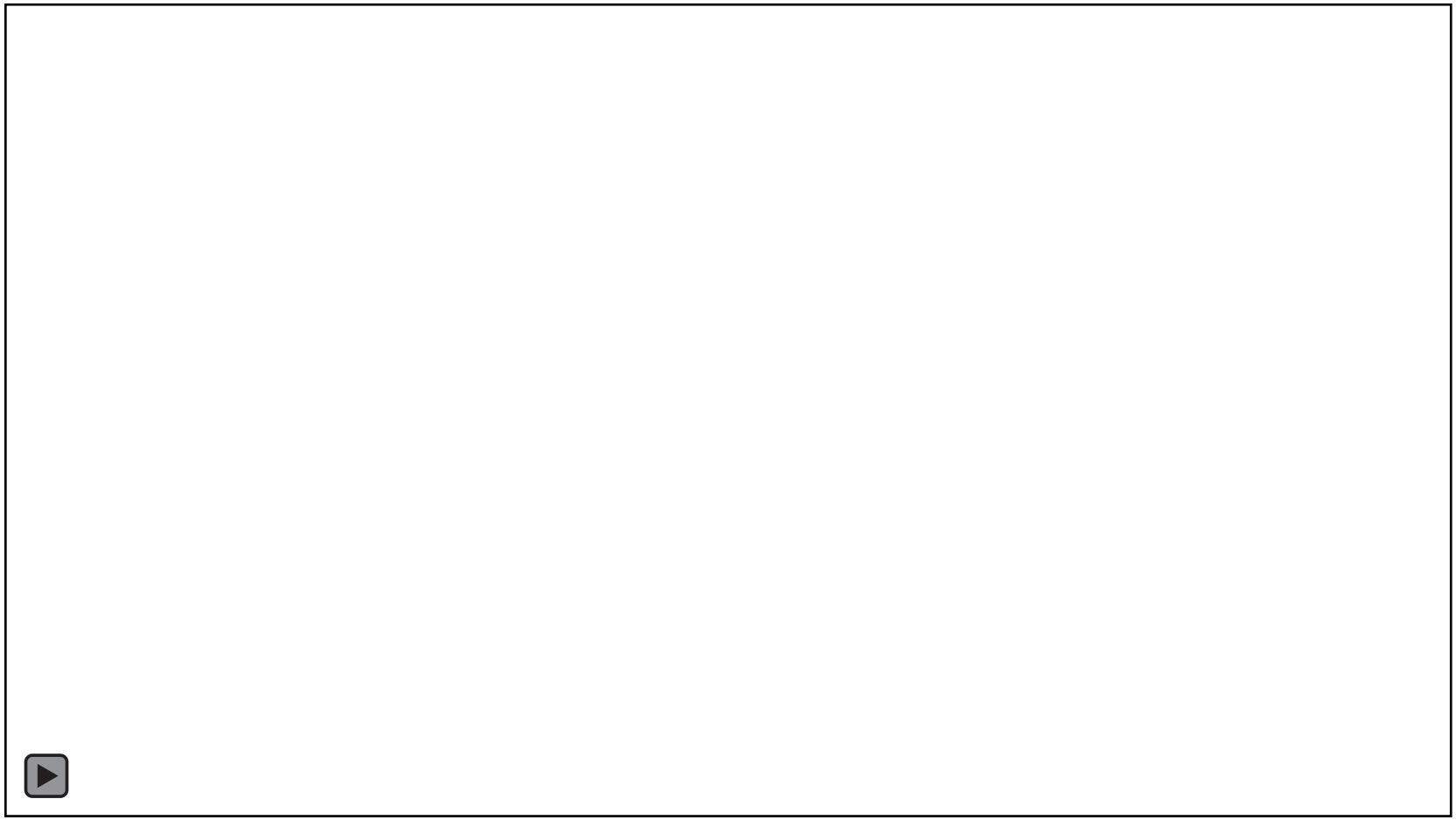
Metodología

CRISP/DM / Data Preparation : EDA



TECNICAS AVANZADAS
(no se usan features)

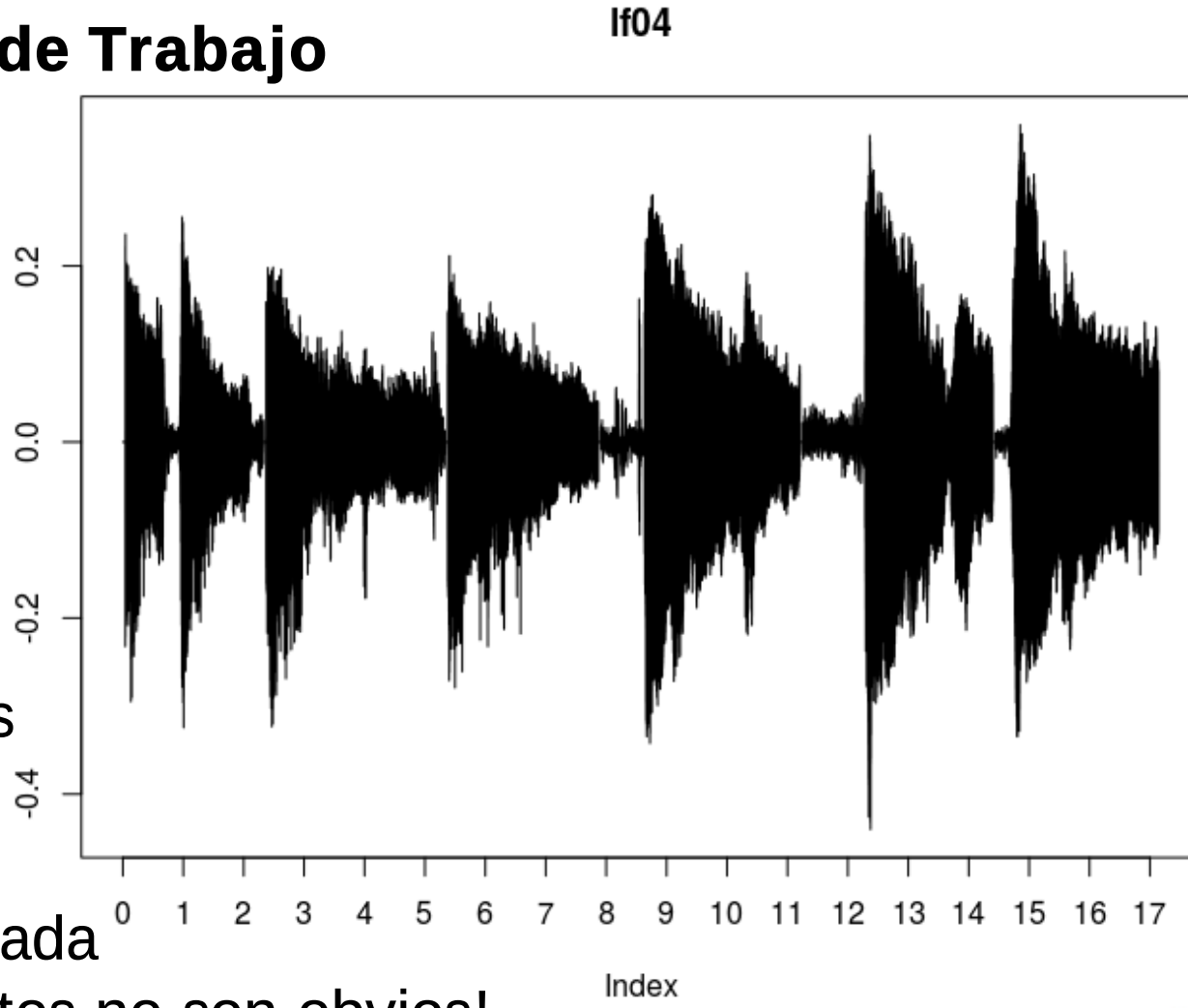
CRISP/DM / Caso de Trabajo



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

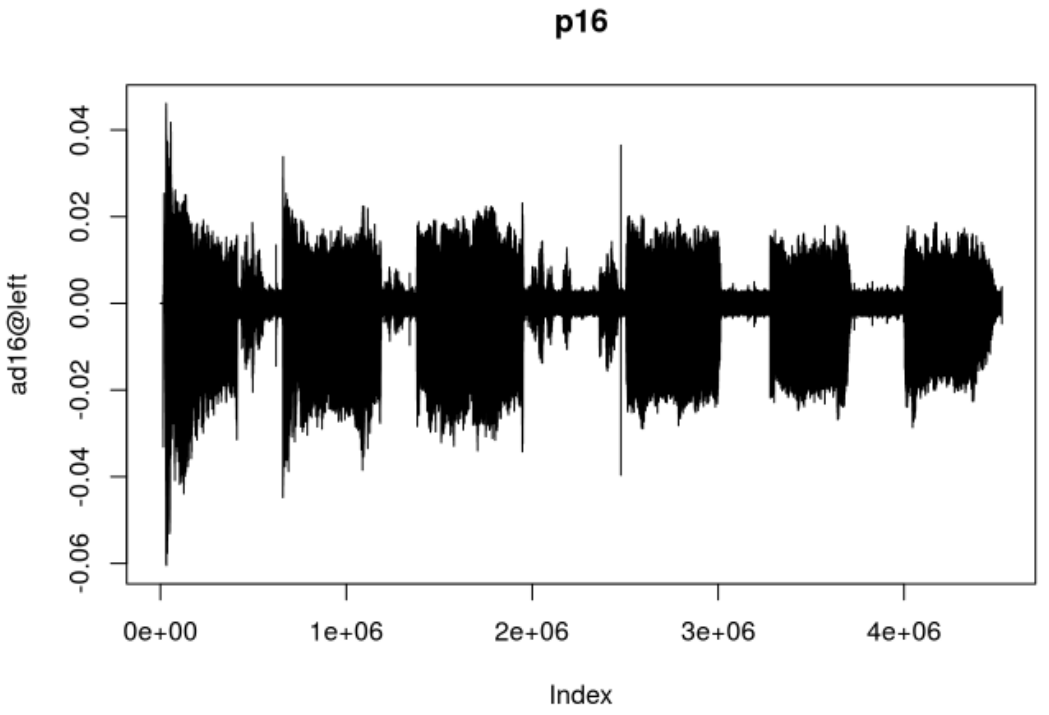
CRISP/DM / Preprocesado

```
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

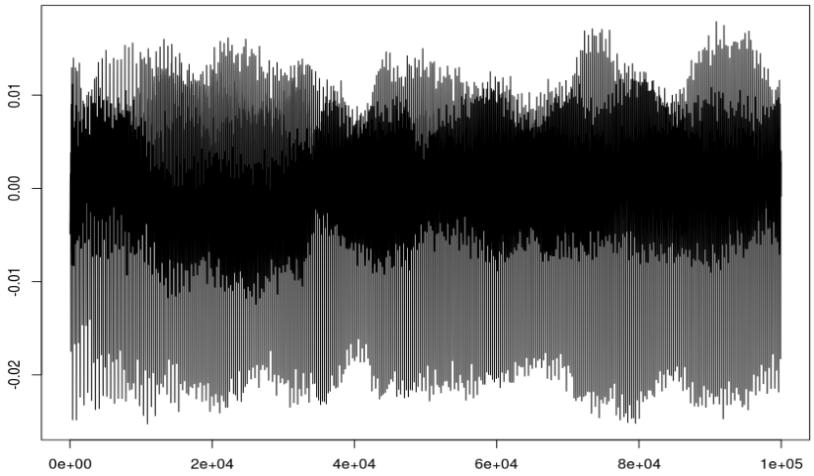
Check the general characters

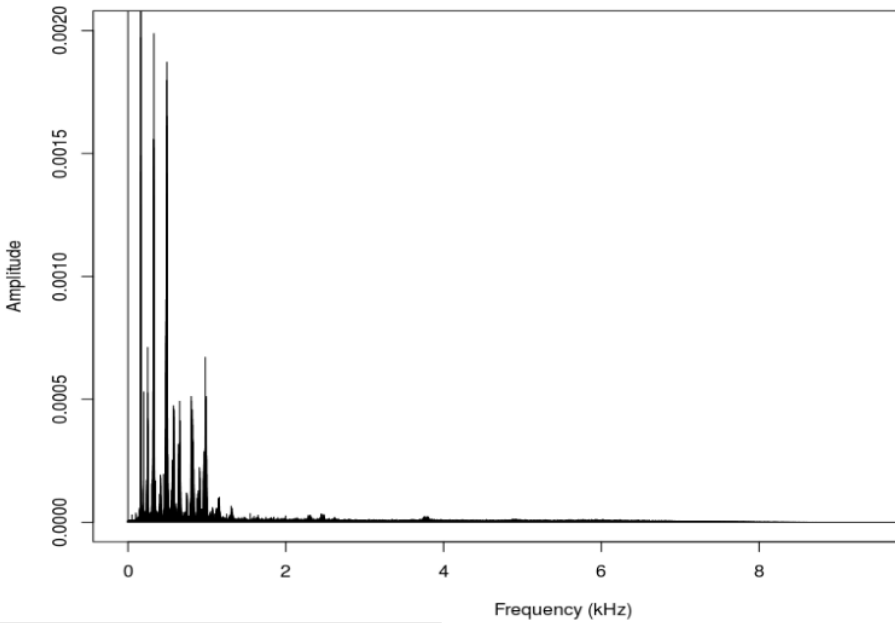
```
###Patient16
setwd('/home/jb/git/DeepLearning/pdaudio')
ad16<-readWave("audio/016.wav")
f16<-ad16@samp.rate
plot(ad16@left,type="l",main="p16")
```

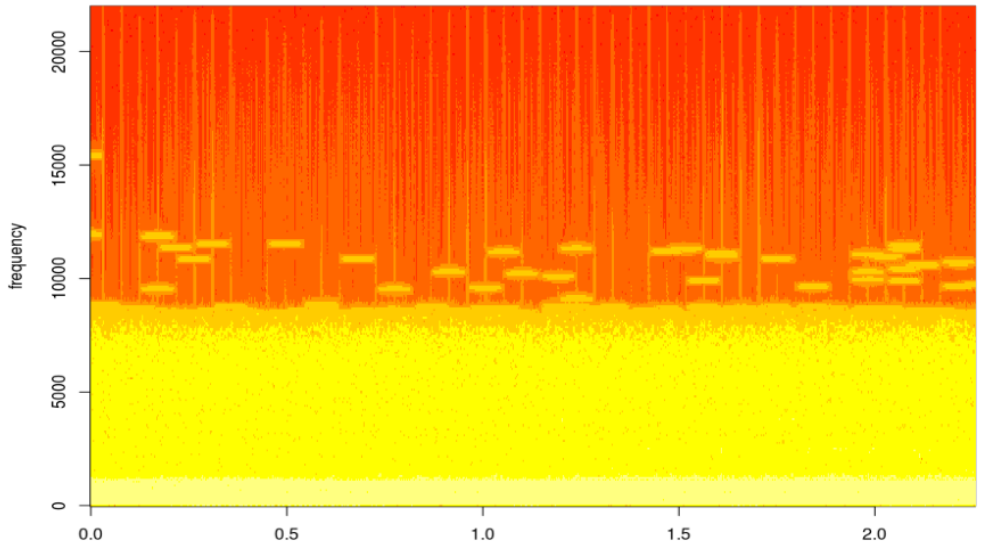


Caso de Trabajo

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	

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

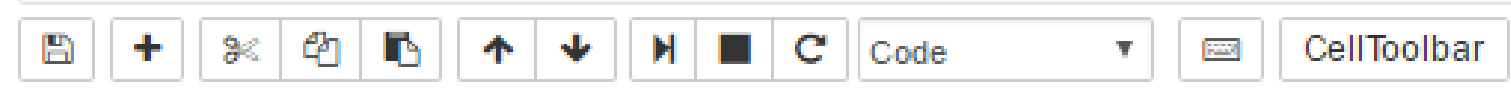
In [1]:
```
import cv2
import numpy as np
import os
from random import shuffle
from tqdm import tqdm
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression
import tensorflow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

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]
```

Caso de Trabajo

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]

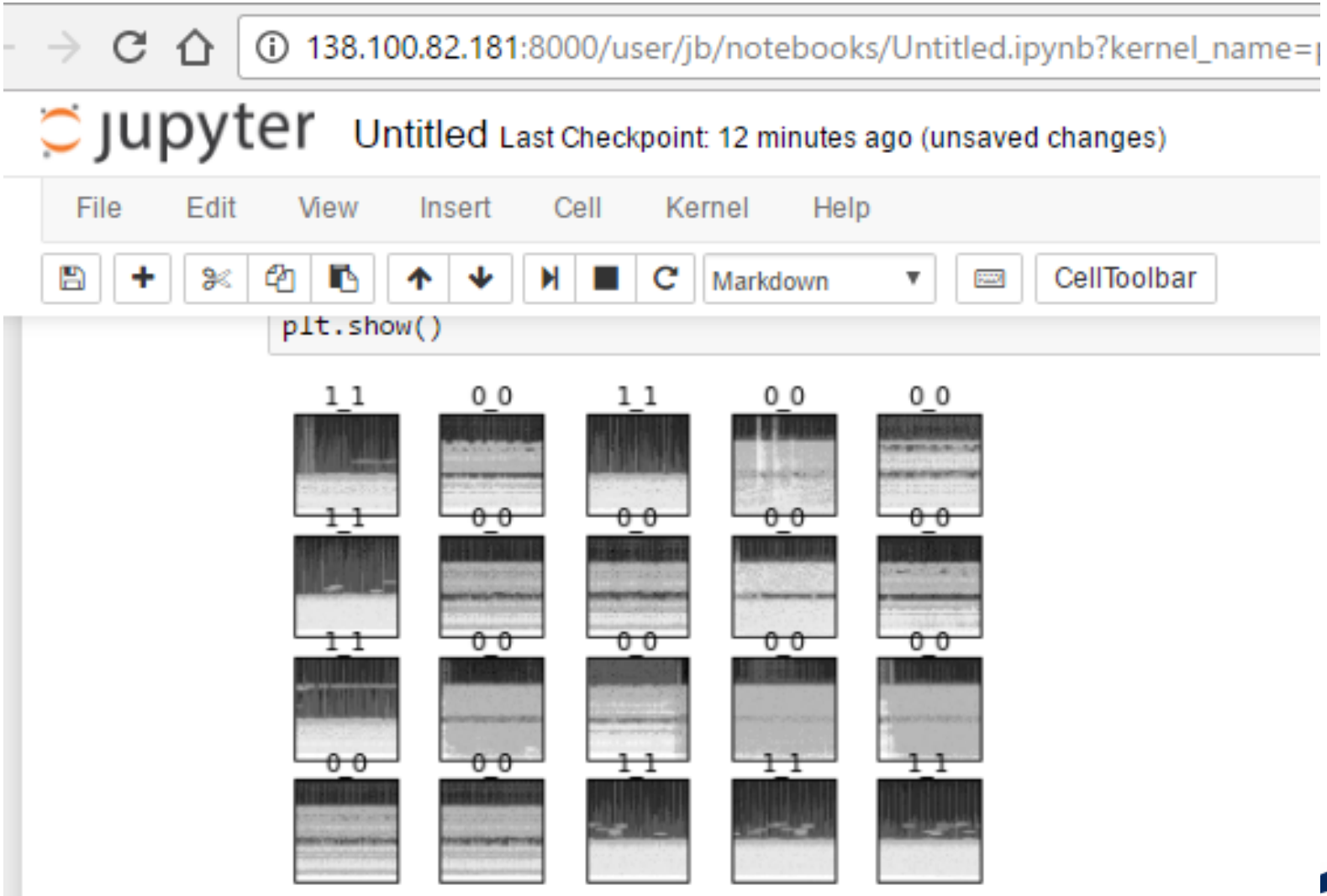
    if np.argmax(model_out) == 1: str_label='1'
    else: str_label='0'

    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

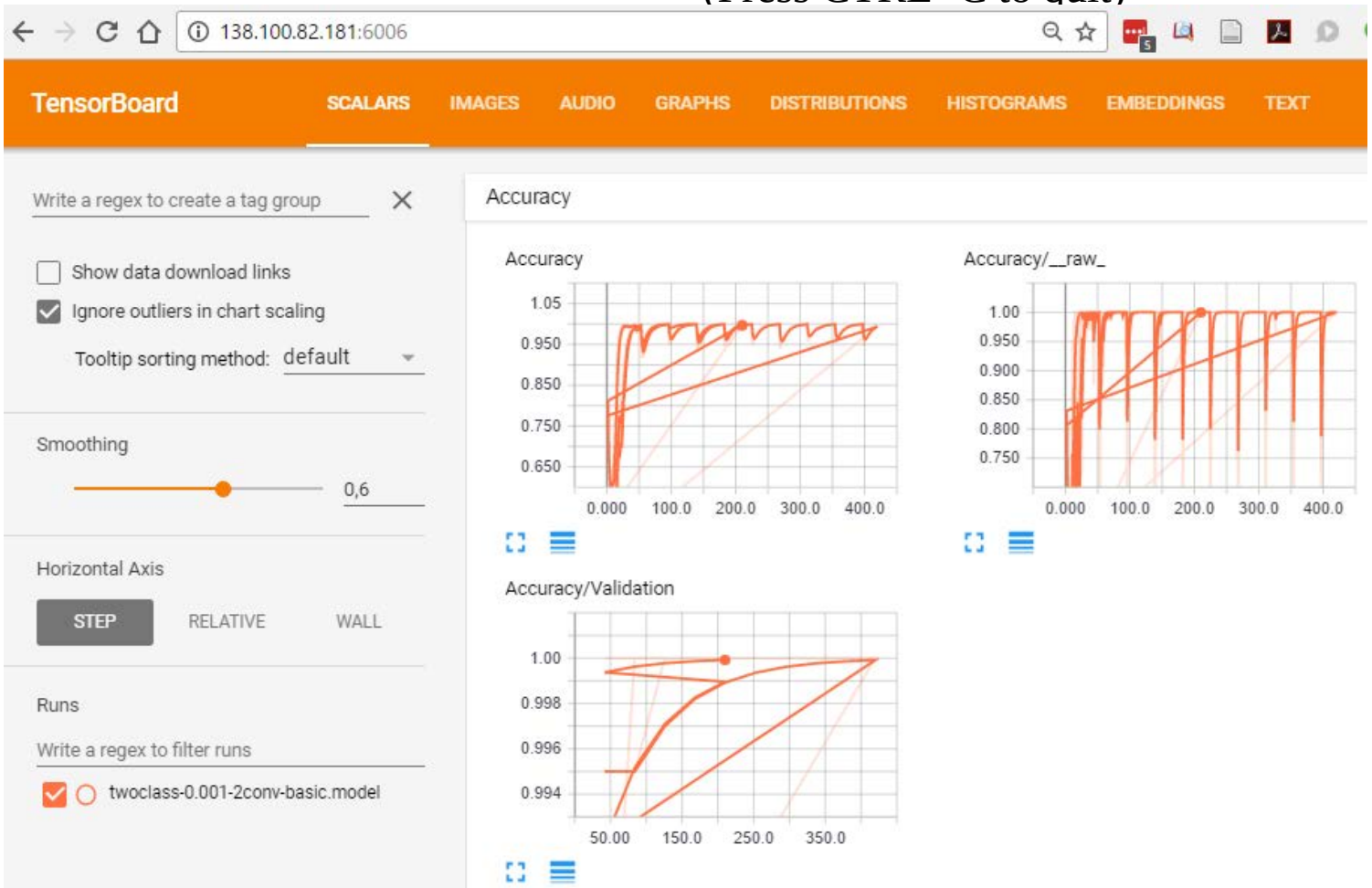


Caso de Trabajo

CRISP/DM / Validación

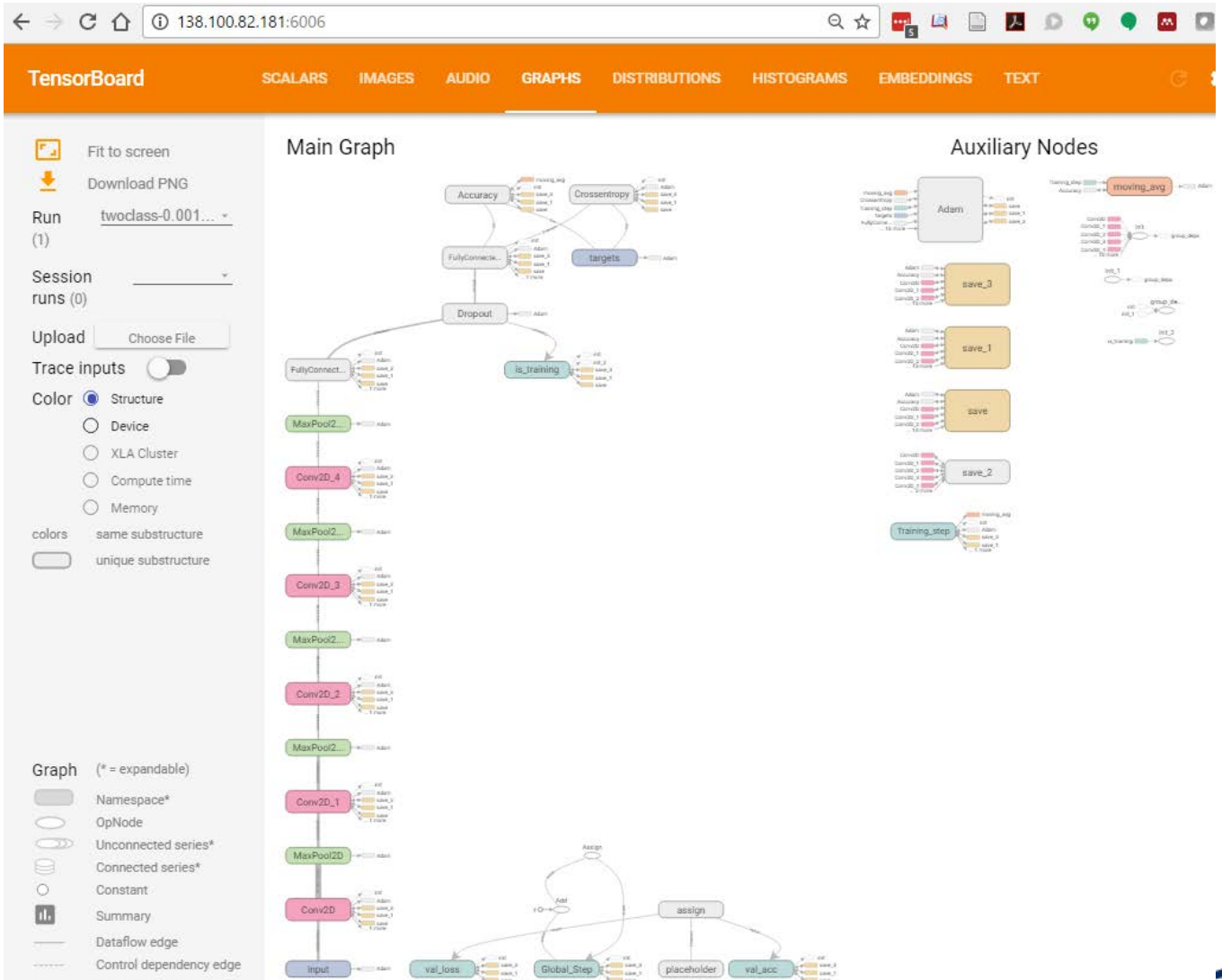
```
$ tensorboard --logdir=log
```

Starting TensorBoard b'47' at http://0.0.0.0:6006
(Press CTRL+C to quit)



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.