Universidad Carlos III de Madrid

SELECCIÓN DE ATRIBUTOS EN R

Fases del análisis de datos

- Recopilación de los datos (tabla datos x atributos)
- Preproceso:
  - De los datos
  - **De los atributos:**
    - **Selección de atributos:**
      - **Ranking**
      - **Subset selection;CFS y WRAPPER**
    - Transformación / Generación de atributos: PCA, random projections
- Generación del modelo:
  - **Clasificación: árboles de decisión**
  - **Regresión:**
    - **Modelos lineales**
    - **Árboles de modelos**
- Evaluación: **validación cruzada, matriz de confusión**
- Despliegue y uso del modelo

Métodos de selección de atributos

Métodos de selección

de atributos

•Ranking (evaluación y ordenación de atributos de manera individual y eliminación de los menos valorados)

•Subset selection (búsqueda del subconjunto de atributos más relevante)

SELECCIÓN ATRIBUTOS EN R

- Fselector package
  - install.packages("FSelector")
  - library(FSelector)

- **http://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Dimensionality_Reduction/Feature_Selection**

RANKING EN R

```
#Carga datos
library(mlbench)
data(HouseVotes84)

#Calcula importancia (peso) de cada atributo
weights <- SOME_FUNCTION(Class~., HouseVotes84)
weights=weights[order(weights$attr_importance,decreasing=TRUE),,drop=F]
print(weights)

#Selecciona los 5 atributos con mas peso (mas importancia)
subset <- cutoff.k(weights, 5)
print(subset)

#Crea un nuevo dataframe con los atributos seleccionados
newHouseVotes = HouseVotes84[,c(subset,"Class")]
head(newHouseVotes )
```

FÓRMULAS PARA RANKING

- Todas intentan ver la correlación entre el atributo y la clase:
- chi.squared(formula, data)
- linear.correlation(formula, data)
- gain.ratio(formula, data)
- …

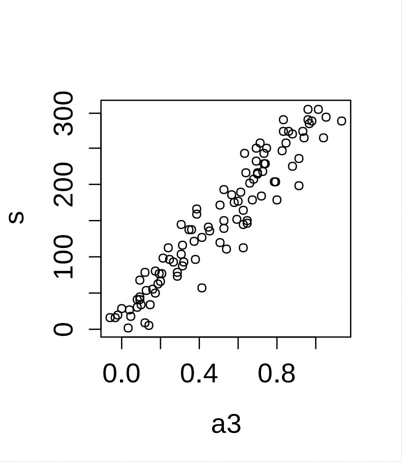
RESULTADO

attr_importance

V4	0.923255954
V3	0.748864321
V5	0.718768923
V12	0.714922593
V8	0.661876085
V9	0.629797943
V14	0.625283342
V13	0.555971176

…

- Class ~ V4 + V3 + V5 + V12 + V8

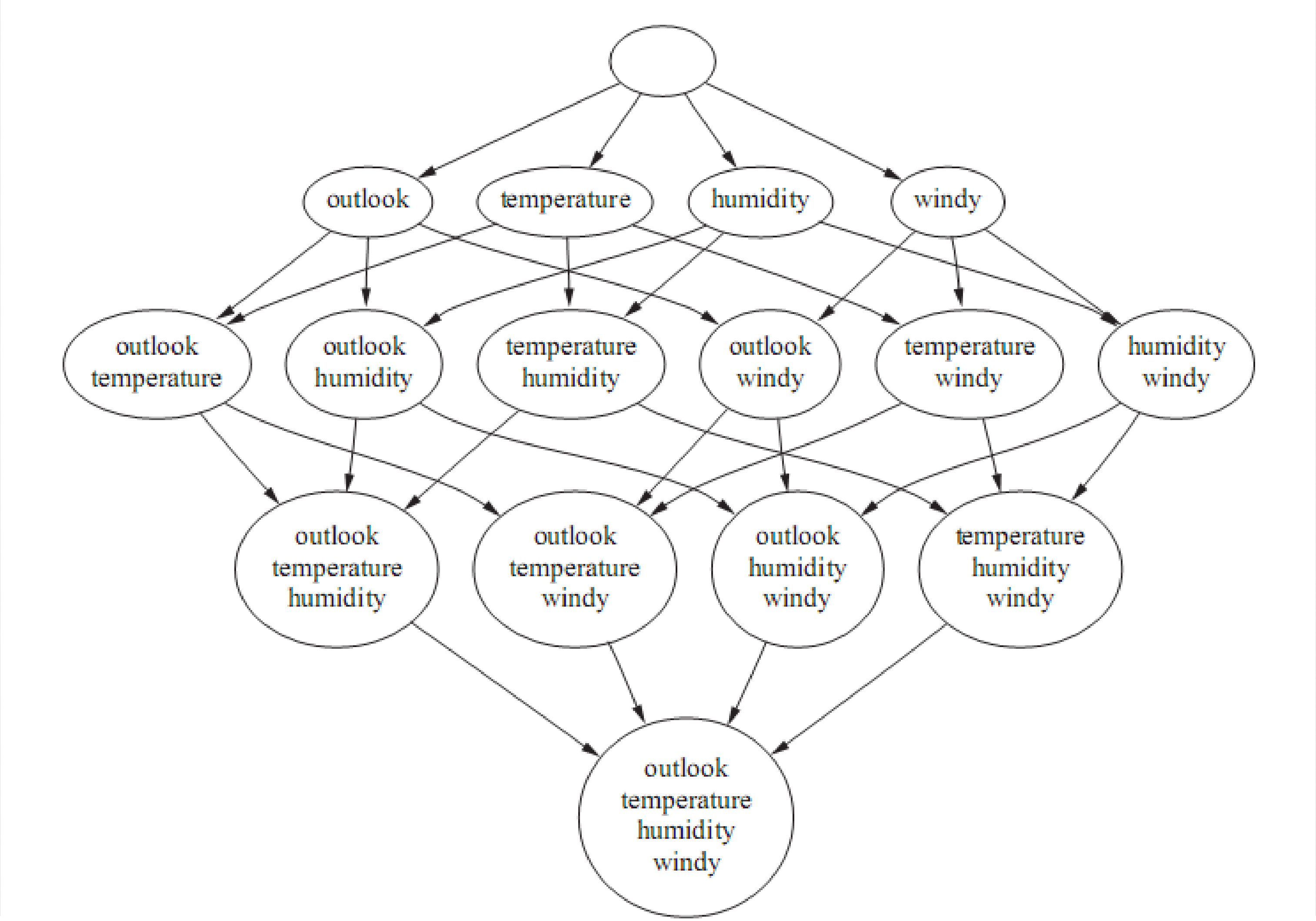


DATAFRAME CON ATRIBUTOS SELECCIONADOS

- head(HouseVotes84[,c(subset,"Class")])

```
V4 V3 V5 V12 V8 Class
1 y n y y n republican
2 y n y y n republican
3 <NA> y y n n democrat
4 n y <NA> n n democrat
5 n y y <NA> n democrat
6 n y y n n democrat
```

Búsqueda en el espacio de **subconjuntos** de atributos



SUBSET SELECTION CON CFS

- subset2=cfs(Class ~ ., HouseVotes84)
- print(subset2)

- [1] "V4“

- Usa best-first search

Evaluación de subconjuntos: Correlation Feature Selection (CFS)

- El método *CFS* evalúa un subconjunto de atributos calculando:
  - La media de las correlaciones (o similar) de cada atributo con la clase
  - Las correlaciones por redundancias entre atributos

$$\text{Evaluación(Ai)} = \frac{\text{correlación con la clase}}{\text{correlaciones entre atributos}} = \sum_j U(A_j, C) \Big/ \sqrt{\sum_i \sum_j U(A_i, A_j)}$$

EJEMPLOS ARTIFICIALES

- Creemos un problema artificial con dos entradas y una salida. La primera entrada está correlacionada con la salida, la segunda no porque es aleatoria

```
a1=1:100
s = 3*(1:100)
a2 =runif(100)
d = data.frame(a1,a2,s)
```

DATAFRAME ARTIFICIAL

	a1	a2	s
1	1	0.34712845	3
2	2	0.08241925	6
3	3	0.77195274	9
4	4	0.51318681	12
5	5	0.75451989	15
6	6	0.71946420	18

weights = chi.squared(s~., d)
weights

Esta es la salida: a1 correlación al 100%, a2 correlación al 0%

attr_importance
a1 1
a2 0

```
weights = information.gain(s~., d)
weights

SALIDA:
attr_importance
a1 2.321928
a2 0.000000
```

AÑADAMOS UN ATRIBUTO REDUNDANTE

```
a3=seq(0,1,length=100)
d=cbind(a3,d)

> head(d)
          a3 a1         a2  s
1 0.00000000  1 0.34712845  3
2 0.01010101  2 0.08241925  6
3 0.02020202  3 0.77195274  9
4 0.03030303  4 0.51318681 12
5 0.04040404  5 0.75451989 15
6 0.05050505  6 0.71946420 18
```

DOS ATRIBUTOS CORRELACIONADOS

```
weights = information.gain(s~., d)
weights
```

- SALIDA: a3 y a1 tienen la misma (alta) importancia, pero obviamente uno de los dos sobra

```
attr_importance
a3 2.321928
a1 2.321928
a2 0.000000
```

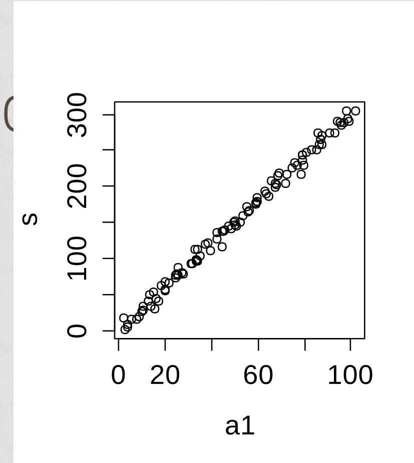
DOS ATRIBUTOS CORRELACIONADOS

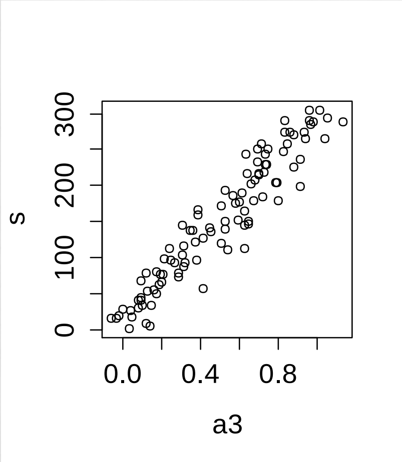
```
subset2=cfs(s ~ ., d)
subset2
[1] "a3“
```

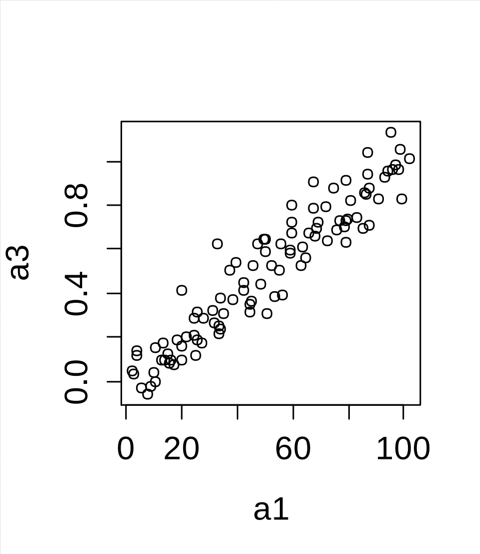
Vemos que se ha quitado de encima los atributos redundantes (a1 en este caso) y los irrelevantes (a2)

OTRO EJEMPLO

```
> a1 = (1:100)+rnorm(100,0,2)
> a3 = seq(0,1,length=100)+rnorm(100,0
> plot(a1,a3)
> a2 = runif(100)
> s = 3*(1:100)+rnorm(100,0,4)
> plot(a1,s)
> plot(a3,s)
> plot(a1,a3)
```







OTRO EJEMPLO

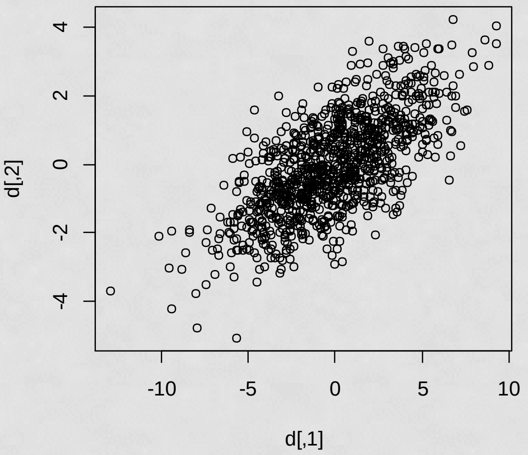
```
> cfs(s~. , data.frame(a1,a2,a3,s))
[1] "a1“

> information.gain(s~., data.frame(a1,a2,a3,s))

attr_importance
a1        2.051957
a2        0.000000
a3        1.429328
```

TRANSFORMACIÓN DE ATRIBUTOS EN R

PCA EN R

- Generación de los datos artificiales

```
install.packages('mnormt')
library(mnormt)
Sigma <- matrix(c(10,3,3,2),2,2)
d=rmnorm(1000,c(0,0),Sigma)
plot(d)
```



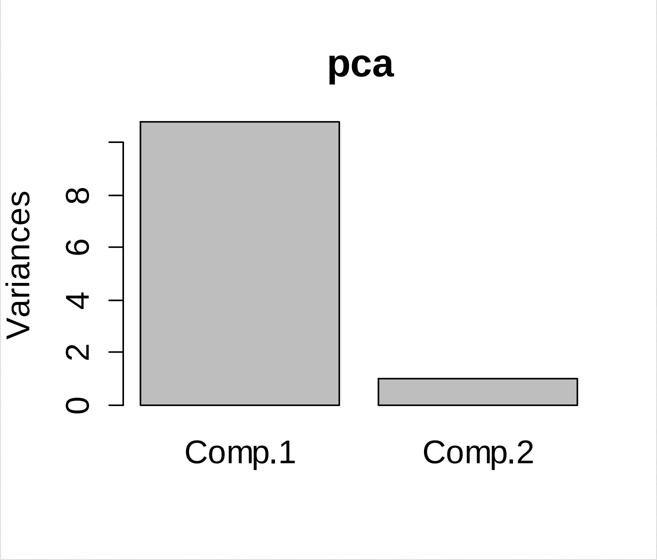
PCA EN R

pca = princomp(d)

describe(pca)

plot(pca)

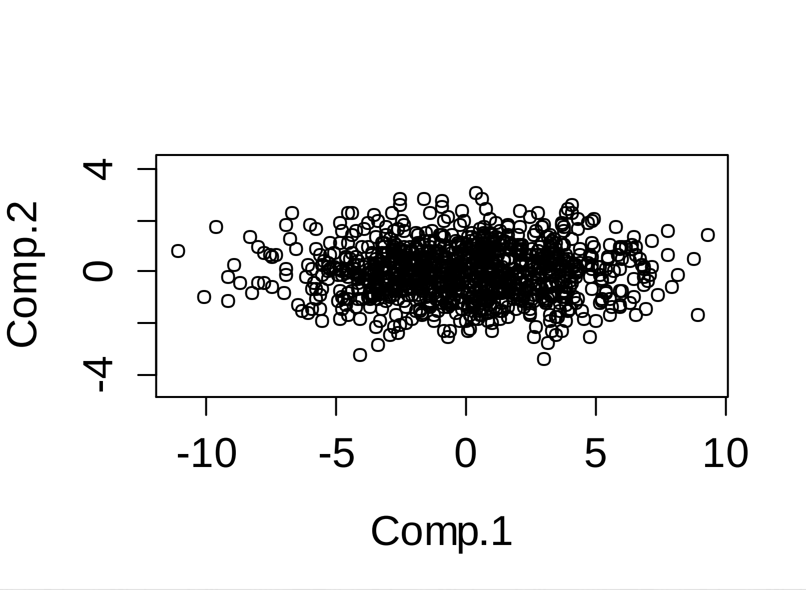
```
summary(pca)
Importance of components:
Comp.1
Standard deviation 3.2830687
Proportion of Variance 0.9135387
Cumulative Proportion 0.9135387
Comp.2
Standard deviation 1.01001369
Proportion of Variance 0.08646126
Cumulative Proportion 1.00000000
```



PCA EN R

- ¿Cómo quedan los datos proyectados?

datosProyectados = pca$scores
plot(datosProyectados)

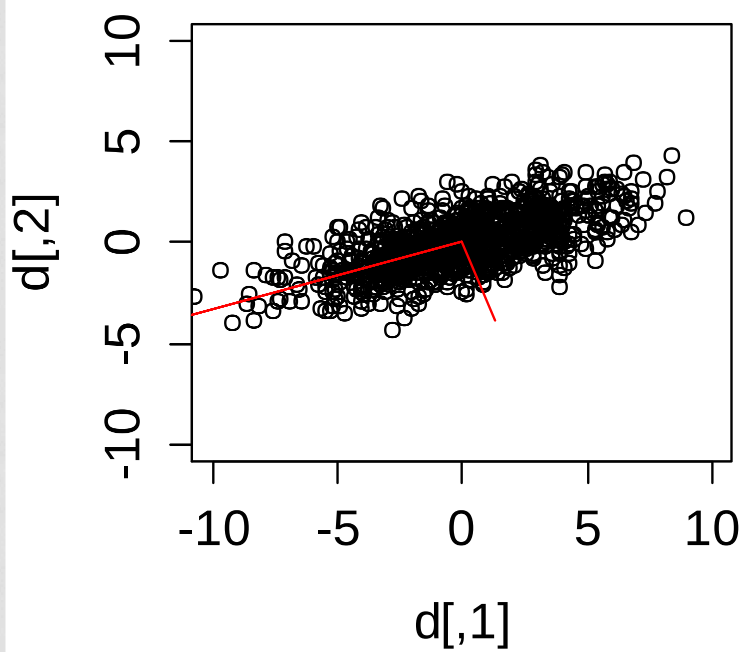


CENTRO Y EJES PCA

```
pca$center
[1] 0.01483700 0.01242027
pca$loadings
Loadings:
      Comp.1 Comp.2
[1,] -0.949 0.315
[2,] -0.315 -0.949
         Comp.1 Comp.2
SS loadings 1.0 1.0
Proportion Var 0.5 0.5
Cumulative Var 0.5 1.0
```

VISUALIZACIÓN DE LOS COMPONENTES

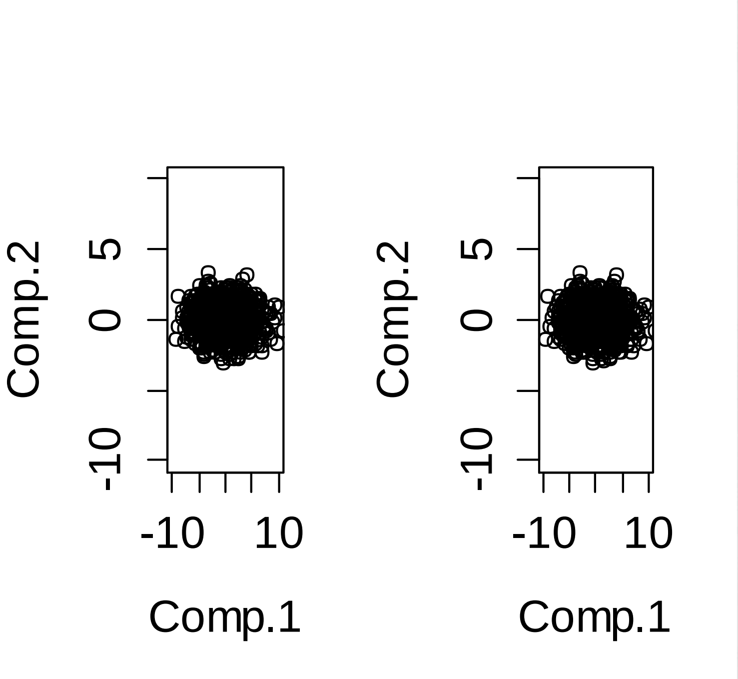
```
v00 =pca$center
v10=pca$center
v01 = pca$loadings[,1]* pca$sdev[1]*4
v11 = pca$loadings[,2]* pca$sdev[2]*4
plot(d,xlim=c(-10,10),ylim=c(-10,10))
m =rbind(v00,v00+v01,v10,v10+v11)
lines(m,col="red")
```



PCA ES UNA TRANSFORMACIÓN LINEAL

- A la izquierda, los datos transformados por pca: pca$scores
- A la derecha, los datos transformados directamente: (d-pca$center) %*% pca$loadings

```
par(mfrow=c(1,2))
plot(pca$scores,xlim=c(-10,10),ylim=c(-10,10))
d_proyectados = d %*% pca$loadings
plot(d_proyectados,xlim=c(-10,10),ylim=c(-10,10))
```



RANDOM PROJECTIONS

```
install.packages("far")
library(far)
(rp = orthonormalization(matrix(rnorm(4),2,2),
norm=T))

[,1] [,2]
[1,]  0.9303622 0.3666417
[2,] 0.3666417 -0.9303622
datos_proyectados_r = d %*% rp
```

RP VS. PCA

plot(datos_proyectados_r,xlim=c(-10,10),ylim=c(-10,10))

plot(pca$scores,xlim=c(-10,10),ylim=c(-10,10))