Machine Learning

Preprocesamiento:
Reducción de dimensionalidad

Dr. Edgar Acuña
Departamento de Ciencias Matemáticas
Universidad de Puerto Rico-Mayaguez

E-mail: edgar.acuna@upr.edu, eacunaf@gmail.com

Website: academic.uprm.edu/eacuna

Reducción de la Dimensionalidad

- ***Selección de variables*:** El principal objetivo de la selección de variables es reducir la dimensionalidad del espacio de variables, seleccionando variables relevantes y no redundantes. Una variable es redundante cuando da información contenida en alguna otra variable. Una variable es irrelevante si da muy poca información. Esto es, la selección de variables selecciona “q" variables del conjunto completo de “p" variables tal que $q \leq p$. Idealmente $q <<< p$.

- ***Extracción de Variables*:** Se construye un conjunto más pequeño de nuevas variables aplicando una transformación linear (o no linear) al conjunto original de variables. El método más conocido es el de Análisis de Componentes Principales (PCA). Otros: PLS, Curvas Principales.

Selección de Variables

Consideraremos solamente el problema de selección de variables para Clasificación Supervisada.

Objetivo: Seleccionar un subconjunto pequeño de variables tal que:

a) La precisión del clasificador en el conjunto de datos no varíe de forma significativa.

b) La distribución condicional resultante de una clase C, dado el vector de variables seleccionado G, esté tan cerca como sea posible a la distribución condicional original dadas todas las variables F.

Ventajas de la Selección de Variables

- El costo computacional de la clasificación será reducido ya que el número de variables será menor que antes.
- La complejidad del clasificador es reducida ya que se eliminan las variables redundantes y las irrelevantes.
- Ayuda a lidiar con el efecto de la "Maldición de la dimensionalidad", que ocurre en procesos que requieren tener un gran número de instancias en comparación con el número de variables.

Pasos en la selección de variables

1. **Proceso de Generación**: La búsqueda del subconjunto óptimo puede ser: completo, heurístico, aleatorio.
2. **Función de Evaluación**: Medidas de distancia, medidas de información, medidas de consistencia, medidas de dependencia o de correlacion, ChiSquare, tasa de error de clasificación.
3. **Criterio de parada**: Un umbral dado, número predefinido de iteraciones, tamaño predefinido del mejor subconjunto de variables.
4. **Proceso de Validación** (Opcional) Verificar si el subconjunto es válido.

Guía para escoger un método de selección de variables

- Habilidad para manejar distintos tipos de variables (continua, binaria, nominal, ordinal)
- Habilidad para manejar múltiples clases.
- Habilidad para manejar grandes conjuntos de datos.
- Habilidad para manejar datos ruidosos.
- Baja complejidad en tiempo.

Categorización de los métodos de selección de variables (Li, Chen y otros 2016)

Medidas de Evaluación	*Generación*		
	Heuristica	*Completa*	*Random*
Distancia	**Relief**	**Branch and Bound**	-
Información	**Trees**	**MDL**	-
Dependencia	**POEIACC**	-	-
Consistencia	**FINCO**	**Focus**	**LVF**
Tasa de Error de Clasificación	**SFS, SBS,SFFS**	**Beam Search**	**Genetic Algorith**

Los métodos de la ultima fila son también conocidos como los métodos “wrapper” (Kohavi, 1997). Los otros son llamados metodos de filtrado.

Determinando visualmente las mejores variables

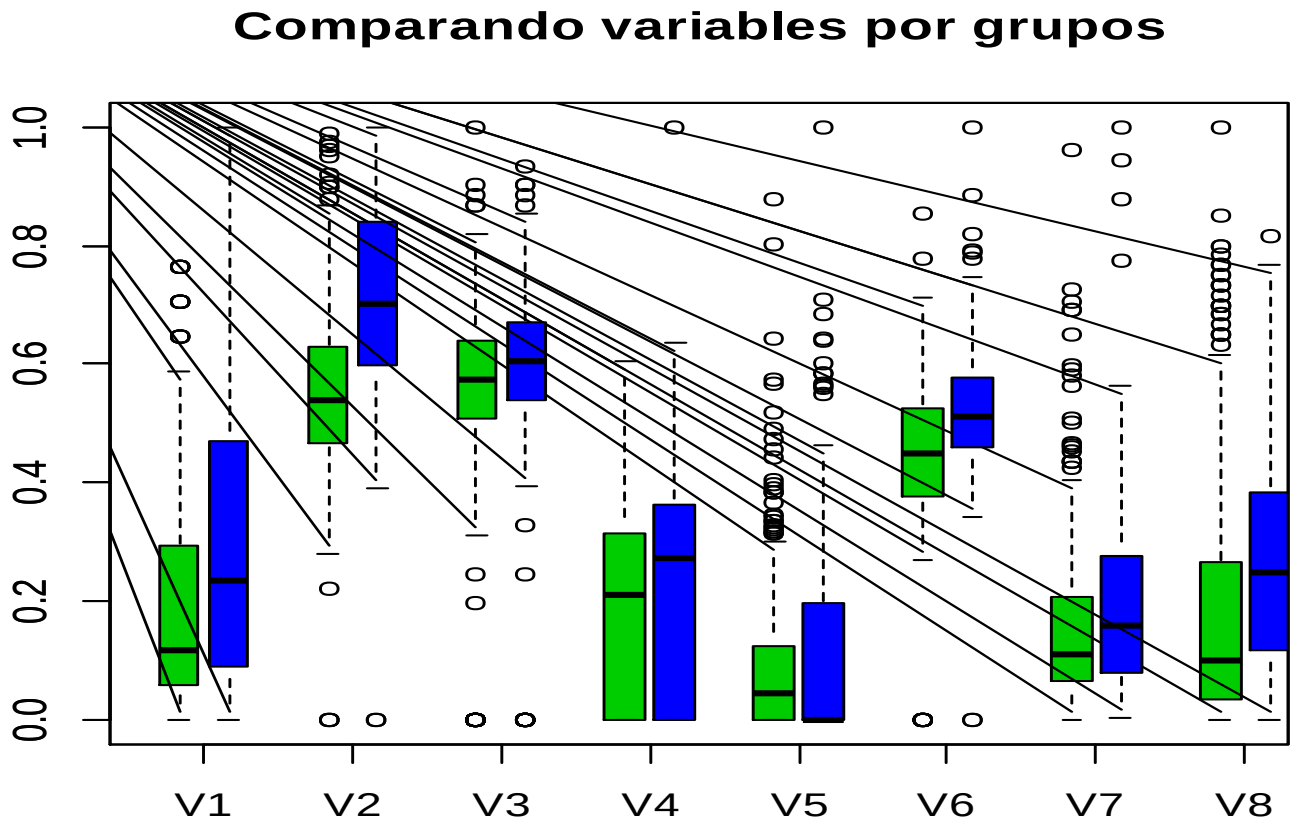
El conjunto de datos Pima Indian Diabetes esta disponible en la UCI Machine Learning Repository. Es un data frame con 768 observations y las siguientes 9 variables:

V1: Number of times pregnant
V2:Plasma glucose concentration (glucose tolerance test)
V3:Diastolic blood pressure (mm Hg)
V4:Triceps skin fold thickness (mm)
V5:2-Hour serum insulin (mu U/ml)
V6:Body mass index (weight in kg/(height in m)\^2)
V7:Diabetes pedigree function
V8:Age (years)
V9:Class variable (1:tested positive for diabetes, 0: tested negative for diabetes)

Determinando visualmente las mejores variables

```
data(diabetes)
#normalizando diabetes
ndiab=mmnorm(diabetes)
a=ndiab[ndiab[,9]==1,]
b=ndiab[ndiab[,9]==2,]
#trazando boxplots comparativos
boxplot(a[,-9],at=0:7*3+1, main="Comparando variables de diabetes",col=
"blue", xaxt="n")
boxplot(b[,-9],at=0:7*3+2,xaxt="n",col= "green", add=T)
axis(1, at = 0:7*3 + 1.5, labels = colnames(a)[-9], tick = TRUE)
```

Variables mas importantes en diabetes



V2,V8, V1,V6 y V7 parecen ser las mas importantes

Usando pruebas estadisticas

En el caso de dos grupos y atributos continuos se puede usar la prueba t y en el caso de mas de dos grupos la prueba F.
En el caso de atributos nominales o categoricos se prueba usar la prueba de Chi-square.
Todas esas pruebas estadisticas asumen propiedades distribucionales de lo datos que raras veces se cumplen, aun asi se siguen usando

```
> acu_F(diabetes)
[1]  39.6702 213.1618   3.2570   4.3044  13.2811  71.7721  23.8713
46.1406
>
```

Las variables mas importantes serian la 2, 6,8,1 y 7.

Métodos de Filtrado

Estos no requieren un clasificador, en lugar de ello usan medidas que permiten seleccionar las variables que distinguen mejor a la clase.

- RELIEF
- Las Vegas Filter (LVF)
- FINCO
- Otros : Branch & Bound, Focus,

El método RELIEF

- Kira and Rendell (1992) para problemas de dos clases y generalizado a problemas multi-clases por Kononenko (1994) y Kononenko, et al. (1997).
- Genera subconjuntos de variables en forma heurística.
- Una variable tiene un peso de relevancia, el cual es grande si puede distinguir claramente dos instancias que pertenecen a distintas clases pero no dos instancias que están en la misma clase.
- Usa una medida de distancia (Euclideana, Manhattan)

El método RELIEF (procedimiento)

- Se selecciona al azar un número dado ***Nsample*** de instancias del conjunto de entrenamiento D que contiene F variables.
- Los pesos de relevancia W_j de cada variable son inicializados en cero.
- Para cada instancia seleccionada **x**, se debe identificar dos instancias particulares:

  Nearhit: La instancia mas cercana a **x** que pertenece a la misma clase.

  Nearmiss: La instancia mas cercana a **x** que pertenece a una clase diferente.

El método RELIEF (cont)

- Luego los pesos W_j's (i=1,..F) son actualizados para cada instancia de la muestra tomada, usando la relación

$$W_j = W_j - diff(x_j, Nearhit_j)^2/NS + diff(x_j, Nearmiss_j)^2/NS$$

Si la variable X_k es nominal o binaria, entonces:

- $diff(x_{ik},x_{jk}) = 1$ para $x_{ik} \neq x_{jk}$

  $= 0$ en caso contrario.

Si la variable X_k es continua u ordinal entonces:

- $diff(x_{ik},x_{jk}) = (x_{ik}-x_{jk})/c_k$, donde $c_k = range(X_k)$

Decisión: Si $W_j \geq \tau$ (un valor preestablecido, usualmente un valor positivo cerca de cero) entonces se selecciona la variable f_j

Programas para RELIEF (cont)

- La librería dprep tiene una funcion relief . Tambien la función attrEval de la librería CORElearn hace ReliefF
- En Rapidminer se elige primero el operador Modelling luego attribute weighting y luego weights for relief.
- El modulo Scikit-learn de Python tiene algunos métodos de Feature selection, pero no tiene el ReliefF.
- En WEKA hay que usar Select Atributes luego como Attribute Evaluator: ReliefFAttributeEval y como Search method: Ranker.
- En dprep solo se usa un vecino mas cercano. Pero Rapidminer, CORElearn y Weka pueden usar mas de un vecino.

Ejemplo de Relief con COREIearn

```
> estReliefF <- attrEval(factor(diabetes$V9) ~ .,
diabetes,estimator="ReliefFequalK", ReliefIterations=100)
> print(estReliefF)
     V1       V2         V3         V4         V5         V6         V7       V8
0.046294 0.145251 0.017579 0.078148 0.028330 0.091047 0.030258 0.077111
>
Atributos elegidos: 2,6,8,4,1
estReliefF <- attrEval(factor(diabetes$V9) ~ .,
diabetes,estimator="ReliefFequalK", kNearestEqual=1,ReliefIterations=100)
> print(estReliefF)
        V1         V2           V3           V4          V5         V6         V7          V8
 0.04313725  0.10435511  0.05841530  0.00542088 -0.00674547  0.04194734
-0.00044122  0.09111111

Atributos elegidos: 2,8,3,1,6
```

Ejemplo de Relief con dprep

```
> relief(diabetes,100,0.005,repet=5)
Features appearing in at least half of repetitions ordered by their average
relevance weight:
     feature frequency    weight
[1,]       1         4 0.0188235
[2,]       8         5 0.0152667
[3,]       2         4 0.0106231
[4,]       7         4 0.0102673
[5,]       6         3 0.0096066
[6,]       3         5 0.0087049
[7,]       4         4 0.0066263
selected features
[1] 1 8 2 7 6 3 4
```

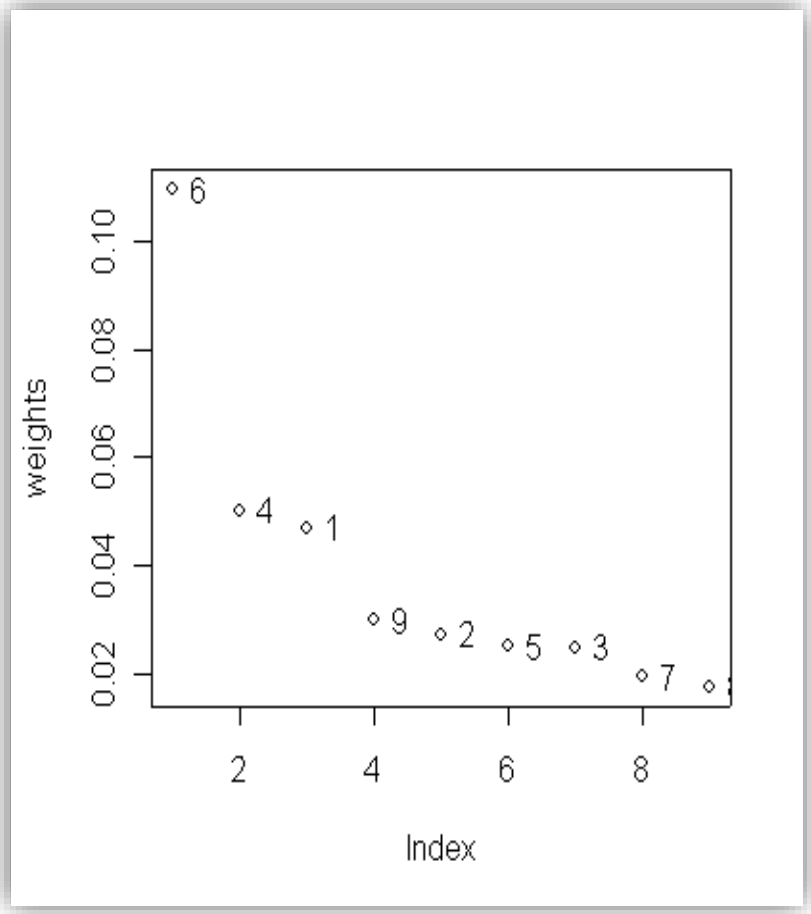
Solo la feature 5 no es seleccionada

Conjunto de datos: Breast-Wisconsin

- 699 instancias, 9 variables y dos clases (benigno o maligno). 16 instancias han sido eliminadas por tener valores faltantes.
- 1. Clump Thickness 2. Uniformity of Cell Size, 3. Uniformity of Cell Shape, 4. Marginal Adhesion, 5. Single Epithelial Cell Size, 6. Bare Nuclei, 7. Bland Chromatin 8. Normal Nucleoli 9. Mitoses.
- Cada variable tiene valores en un rango de 0 a 10.

Ejemplo de Relief: Breastw

```
> relief(breastw,600,0)
      Variables que aparecen en por lo menos la mitad de las
      repeticiones ordenadas por su peso de relevancia
      promedio:
     feature frequency     weight
[1,]       6        10 0.10913169
[2,]       4        10 0.05246502
[3,]       1        10 0.04682305
[4,]       9        10 0.03171399
[5,]       2        10 0.02869547
[6,]       3        10 0.02566461
[7,]       5        10 0.02512963
[8,]       7        10 0.02096502
[9,]       8        10 0.01708025
Variables Seleccionadas
[1] 6 4 1 9 2 3 5 7 8
> relief(breastw,600,0.04)
      Variables que aparecen en por lo menos la mitad de las
      repeticiones ordenadas por su peso de relevancia
      promedio:
feature frequency     weight
[1,]       6        10 0.10844239
[2,]       4        10 0.05293210
[3,]       1        10 0.04853909
Variables Seleccionadas
[1] 6 4 1
```



Ejemplo de Relief Breastw

```
> estReliefF <- attrEval(factor(breastw$V10) ~ .,
breastw,estimator="ReliefFequalK", ReliefIterations=300)
> print(estReliefF)
  V1       V2       V3       V4       V5       V6       V7       V8       V9
0.34960 0.50020 0.51060 0.18620 0.28620 0.57236 0.36780 0.32480 0.05060
Atributos elegidos: 6,3,2,7,1,8,5
estReliefF <- attrEval(factor(breastw$V10) ~ .,
breastw,estimator="ReliefFequalK", kNearestEqual=1,ReliefIterations=300)
> print(estReliefF)
     V1       V2       V3       V4       V5       V6       V7       V8
0.433333 0.193333 0.180000 0.100000 0.110000 0.603105 0.490000 0.416667
     V9
0.073333

Atributos elegidos: 6,7,1,8
```

Método Relief: problema multi-clase

Primero se debe encontrar un ***Nearmiss*** por cada clase distinta a la que pertenece **x**, y luego se promedia su contribución usando los pesos basados en anteriores. Los pesos son actualizados usando:

$$W_j = W_j - diff(x_j, \text{Nearhit})^2 + \sum_{C \neq class(x_j)} \frac{P(C)}{1 - P(class(x_j))} diff(x_j, \text{Nearmiss}(C))^2$$

Conjunto de datos: Landsat

- 6435 instancias, 36 predictoras continuas y 6 Clases

(suelo rojo, cultivo de algodon, suelo gris, suelo gris humedo, suelo con vegetacion rastroja, suelo gris bien humedo)

```
> relief(Satellite,1500,0,1)
Features appearing in at least half of repetitions ordered by their average
relevance weight:
      feature frequency     weight
 [1,]      25         1 0.12518576
 [2,]      21         1 0.12121374
 [3,]      13         1 0.12050074
 [4,]      17         1 0.11886385
 [5,]      33         1 0.11115147
 [30,]      23         1 0.04898387

……

[31,]      31         1 0.04776801
[32,]      12         1 0.04584572
[33,]      19         1 0.04535417
[34,]      11         1 0.04424004
[35,]       7         1 0.04287936
[36,]       8         1 0.04203942
selected features
 [1] 25 21 13 17 33 29  9  5  1 26 14 22 18 30  2 34 10 36  3 16 27  6 20 28  4
[26] 35 24 15 32 23 31 12 19 11  7  8
```

El método Relief (Cont)

Ventajas:

Trabaja bien con variables ruidosas y correlacionadas.
La complejidad en tiempo es linear en el número de variables y en número de muestras tomadas (Nsample).
Trabaja para cualquier tipo de variable.

Desventajas:

Elimina las variables irrelevantes pero no elimina las variables redundantes.

Elección del umbral.
Elección de Nsample.

Método Las Vegas Filter (LVF)

Liu y Setiono (1997)

- El subconjunto de variables es elegido al azar.
- La función de evaluación usada es una medida de inconsistencia.
- Dos instancias son inconsistentes si tienen los mismos valores en las variables pero pertenecen a clases diferentes.
- Las variables continuas del conjunto de datos tienen que ser previamente discretizadas.

Medida de Inconsistencia

La inconsistencia de un conjunto de datos con solo variables no-continuas está dado por:

$$\frac{\sum_{i=1}^{K} (|D_i| - h_i)}{N}$$

K: Número de combinaciones distintas de las N instancias.

$|D_i|$: Cardinalidad de la i-ésima combinación.

h_i: Frecuencia de la clase modal en la i-ésima combinación.

Inconsistencia: Ejemplo

```
> m1
     col1 col2 col3 col4 class
[1,]  1    2  2    1    1
[2,]  4    3  2    2    2
[3,]  4    3  2    2    1
[4,]  1    3  8    1    1
[5,]  9    3  8    2    2
[6,]  9    3  8    1    2
[7,]  9    3  1    2    1
> inconsist(m1)
[1] 0.1428571
```

Algoritmo LVF

Input : D = conjunto de datos , p = número de variables,
S = conjunto de todas las variables,
MaxTries = número máximo de ensayos,
Umbral = τ .
$C_{best} = p$, $S_{best} = S$
Para desde i= 1 hasta MaxTries
S_i = Subconjunto de S elegido al azar.
C = card(S_i)
Si ($C < C_{best}$)
{ Si Inconsistencia (S_i, D) $<\tau$
$S_{best} = S_i$, $C_{best} = C$ }
si ($C = C_{best}$ y Inconsistencia (S_i, D)$\leq \tau$)
$S_{best} = S_i$.
Output : S_{best}

Ejemplos

> dbupa=disc.ew(bupa,1:6)
>inconsist(dbupa)
[1] 0.01159420
> lvf(dbupa,.1,1000)
La inconsistencia del mejor subconjunto es:
0.05217391
El mejor subconjunto de variables es:
[1] 1 2 3 6
> lvf(breastw,.01,2000)

La inconsistencia del mejor subconjunto es:

0.005856515

El mejor subconjunto de variables es:

[1] 1 6 8

Desventajas de LVF

- Selección del umbral. Un umbral con un valor pequeño implicará la selección de un número grande de variables.
- Un número grande de iteraciones disminuye la variabilidad de conjunto seleccionado pero hace lento el cómputo.

Método FINCO

FINCO (Acuna, 2002) combina la selección secuencial hacia adelante con una medida de inconsistencia como función de evaluación.

PROCEDIMIENTO

- Se inicializa el mejor subconjunto de variables T como un conjunto vacío.
- En el primer paso, se selecciona la variable que produce el menor nivel de inconsistencia.
- Luego, se selecciona la variable que junto con la primera variable seleccionada produzca el menor nivel de inconsistencia.
- El proceso continúa hasta que cada variable que aun no fue seleccionada junto con las variables que ya estén en T produzcan un nivel de inconsistencia menor que un umbral τ.

Algoritmo FINCO

Input : D = Conjunto de datos, p = Número de variables en D,
S = conjunto de todas las variables, Umbral = τ .

Initialization:

Set k=0 and $T_k = \phi$

Inclusión: Para k=1 hasta p

Seleccionar variable x^+ tal que:

$$x^+ = \arg\min_{x \in S - T_k} Incons(T_k + x)$$

donde S- T_k es el subconjunto de variables que aun no fueron seleccionadas.

si $Incons(T_k+x^+) < Incons(T_k)$ and $Incons(T_k+x^+) > \tau$, entonces

$T_{k+1} = T_k + x^+$ y k:=k+1

sino parar

Output: T_k: subconjunto de variables seleccionadas

Ejemplos

```
> finco(dbupa,.05)
Variables seleccionadas y sus tasas de inconsistencia
$varselec
[1] 2 1 6 3
$inconsis
[1] 0.37681159 0.26376812 0.13333333 0.05217391
> finco(breastw,.01)
Variables seleccionadas y sus tasas de inconsistencia
$varselec
[1] 2 6

$inconsis
[1] 0.07027818 0.02635432
```

finco(breastw,.001)
Variables seleccionadas y sus tasas de inconsistencia
$varselec
[1] 2 6 1

$inconsis
[1] 0.070278184 0.026354319 0.005856515

- El umbral se elige como un valor un poco mas grande que la inconsistencia del conjunto de datos, para evitar quedarse con todas las variables originales..

LVF se puede hacer en WEKA usando
ConsistencySubsetEval como attribute evaluator
y como search method RandomSearch.
FINCO tambien se puede hacer en WEKA usando
ConsistencySubsetEval como attribute evaluator
y como search method BestFirst.

Métodos Wrapper

Wrappers usan la tasa de error de mala clasificación, obtenida con un clasificador dado, como la función de evaluación para el subconjunto de variables.

- Selección secuencial hacia adelante (SFS)
- Selección secuencial hacia atrás (SBS)
- Selección secuencial flotante hacia adelante (SFFS)
- Otros: SFBS, Take l-remove r, Recursive Feature Elimination, GSFS, GA, SA.

Selección secuencial hacia adelante (SFS)

- Se inicializa el mejor subconjunto de variables T como el conjunto vacío.
- La primera variable que ingresa a T es aquella con la tasa de prediccion más alta con un clasificador dado.
- La segunda variable que ingresa a T será aquella que junto con la variable seleccionada en el paso previo produzca la tasa de prediccion más alta.
- El proceso continúa y en cada paso ingresa solamente una variable a T hasta que la tasa de prediccion no crezca cuando el clasificador se construya usando las variables que ya estén en T mas cada una de las demás variables.

Ejemplos: Diabetes y Breastw

Los paquetes dprep y mlr hacen selección usando wrappers:

Primero mostramos con el dprep y eliminando la variable irrelevante V5 de diabetes

```
> sfs(diabetes[,-5],method="knn",kvec=5,repet=10)
The best subset of features is:
[1] "V2" "V6" "V8"
> sfs(diabetes[,-5],method="rpart",repet=10)
The best subset of features is:
[1] "V2" "V6" "V7"
> sfs(diabetes[,-5],method="lda",repet=10)
The best subset of features is:
[1] "V6" "V2" "V7“
```

Eliminando las varaibles irrelevantes V4 y V9 de breastw

```
> dim(breastw)
[1] 699  10
> breastw1=na.omit(breastw) #eliminando las filas con missings
> dim(breastw1)
[1] 683  10
```

Ejemplos: Diabetes y Breastw

```
> sfs(breastw1[,-c(4,9)],"knn")#Knn con 5 vecinos
The best subset of features is:
[1] "V6" "V3" "V7" "V5"
> sfs(breastw1[,-c(4,9)],"rpart")
The best subset of features is:
[1] "V6" "V5" "V3"
> sfs(breastw1[,-c(4,9)],"lda")
The best subset of features is:
[1] "V6" "V2" "V1" "V8"
Usando  la librería mlr: Selección de variables para breastw estimando la
    tasa de predicción por Validacion Cruzada
breastw1.task=makeClassifTask(data = breastw1, target ="V10")
rdesc = makeResampleDesc("CV")
 ctrl = makeFeatSelControlSequential(method = "sfs", maxit = NA)
res = selectFeatures("classif.rpart", breastw1.task, rdesc, control = ctrl)
analyzeFeatSelResult(res)
 Features         : 2
Performance      : mmce.test.mean=mmce.test.mean=0.04546
V3, V6
```

Selección secuencial hacia atrás (SBS)

- Inicialmente el mejor subconjunto de variables T incluyen todas las variables del conjunto de datos.
- En el primer paso, se realiza la clasificación sin considerar cada una de las variables, y se elimina la variable donde la tasa de reconocimiento es la más baja.
- El proceso continúa eliminando una variable en cada paso hasta que las tasas de reconocimiento con las variables usadas comienzan a decrecer.

  No eficiente para los clasificadores no paramétricos porque tiene un alto tiempo de computación

Selección secuencial hacia atrás (SBS)

Usando la librería mlr: Selección de variables para breastw estimando la tasa de predicción por Validacion Cruzada

```
breastw1.task=makeClassifTask(data = breastw1[.-c(4.9)], target ="V10")
rdesc = makeResampleDesc("CV")
 ctrl = makeFeatSelControlSequential(method = "sbs", maxit = NA)
res = selectFeatures("classif.rpart", breastw1.task, rdesc, control = ctrl)
analyzeFeatSelResult(res)
> analyzeFeatSelResult(res)
Features         : 3
Performance      : mmce.test.mean=0.04239
V3, V5, V6
Path to optimum:
- Features:    7  Init   :                 Perf = 0.055627  Diff: NA  *
- Features:    6  Remove : V8              Perf = 0.052664  Diff: 0.0029625 *
- Features:    5  Remove : V2              Perf = 0.052664  Diff: 0  *
- Features:    4  Remove : V7              Perf = 0.048252  Diff: 0.0044118 *
- Features:    3  Remove : V1              Perf = 0.042391  Diff: 0.005861  *
```

Selección secuencial flotante hacia adelante(SFFS)

Pudil, et al (1994). Trata de resolver el problema de anidamiento que aparece en SFS y SBS.

- Inicialmente el mejor subconjunto de variables T es el conjunto vacio.
- En cada paso se incluye una nueva variable en T usando SFS, pero es seguida por una verificación de una posible exclusión de variables que ya estén en T. Las variable son excluidas usando SBS hasta que la tasa de reconocimiento comience a decrecer.
- El proceso continúa hasta que SFS no se pueda ejecutar.

Ejemplos

```
sffs(breastw,“rpart")
The selected features are:
[1] "V3" "V6" "V5“

breastw1.task=makeClassifTask(data = breastw1[,-c(4.9)], target ="V10")
 ctrl = makeFeatSelControlSequential(method = "sffs", maxit = NA)
res = selectFeatures("classif.rpart", breastw1.task, rdesc, control = ctrl)
analyzeFeatSelResult(res)
Features         : 3
Performance      : mmce.test.mean=0.04092
V2, V5, V6

Path to optimum:
- Features:    0  Init   :                Perf = 0.43615  Diff: NA  *
- Features:    1  Add    : V2                 Perf = 0.084889  Diff: 0.35126  *
- Features:    2  Add    : V6                 Perf = 0.040921  Diff: 0.043968  *

Stopped, because no improving feature was found.
```

Recursive Feature Elimination

Es un método de selección de variables del tipo de eliminación hacia atrás introducido por Guyon, et al (2002) donde el clasificador usado es el Support vector machine. Pero luego ha sido usando con otros clasificadores como random Forest y hasta con regresión lineal. La importancia de una variable es determinada por el valor de su coeficiente.

La librería caret de R hace RFE.

La función RFE del modulo scikit learn de Python es mucho mas fácil de usar(ver mi github)

Métodos Hibridos

Son una combinacion de lo metodos de filtrado y “wrappers”. El ejemplo tipico es el metodo de Maxima relevancia y Minima Redundancia mRMR (Peng, Long and Ding, 2005)

Primero se determina las variables mas relevantes usando pruebas estadisticas de T o de F cuando los atributos son continuos o Informacion Mutua si los atributos son discretos. Por supuesto, los atributos con los valores mas altos son los mas relevantes. y luego se eliminan las variables redundantes usando diferencias(MID) o cocientes (MIQ) .

Dprep hace mRMR

```
> mRMR_cont(diabetes,4,"MIQ")
[1] 2 6 1 7
> mRMR_cont(diabetes,4,"MID")
[1] 2 6 8 1
> mRMR_disc(breastw,5,"MIQ")
[1] 2 3 7 8 9
> mRMR_disc(breastw,5,"MID")
[1] 2 9 8 6 7
```

Conclusiones

- Entre los métodos wrapper, SFFS se comporta mejor que SFS: menor porcentaje de variables seleccionadas y casi la misma exactitud que SFS. Cómputo rápido.
- Entre los métodos de filtrado, FINCO tiene el menor porcentaje de variables seleccionadas.
- El rendimiento de LVF y RELIEF es bastante similar, pero LVF toma más tiempo de cómputo.
- Los métodos wrapper son más efectivos que los metodos de filtrado reduciendo el error de mala clasificación.
- La velocidad de cómputo de los métodos de filtrado es afectada por el tamaño de la muestra y el número de clases.

Conclusiones (Cont.)

- SFFS y FINCO tienen menor porcentaje de variables seleccionadas .
- En LVF, un incremento en el número de iteraciones disminuye la variabilidad de las variables seleccionadas.
- En LVF y FINCO, una reducción del nivel de inconsistencia mínimo aumenta el número de variables seleccionadas.

Libreria en Python para seleccion de variables

Disponible en http://featureselection.asu.edu/.
Tiene alrededor de 30 algoritmos para seleccion de variables, incluyendo Relief, F-score, Chi-Square, Ganancia de Informacion, mRMR y wrappers. Pero debe ser usado con precaucion por pequenos defectos.

Extracción de Variables

Consideraremos solamente problemas de Clasificación Supervisada.

Objetivo: Construir nuevas variables a partir de las variables originales tal que la precisión del clasificador construido usando estas nuevas variables originales no varíe de forma significativa.

Veremos solamente el metodo de componentes principales.

Análisis de Componentes Principales (PCA)

El objetivo del Análisis de Componentes Principales (Hotelling, 1933) es reducir la información disponible. Esto es, la información contenida en p variables $\mathbf{X}=(X_1,\ldots,X_p)$ puede ser reducida a $\mathbf{Z}=(Z_1,\ldots Z_q)$, *con* $q<p$. Las nuevas variables Z_i's , son llamadas ***Componentes Principales*** y no son correlacionadas.

Los componentes principales de la matriz **X** son los elementos de una transformación lineal ortogonal de **X**

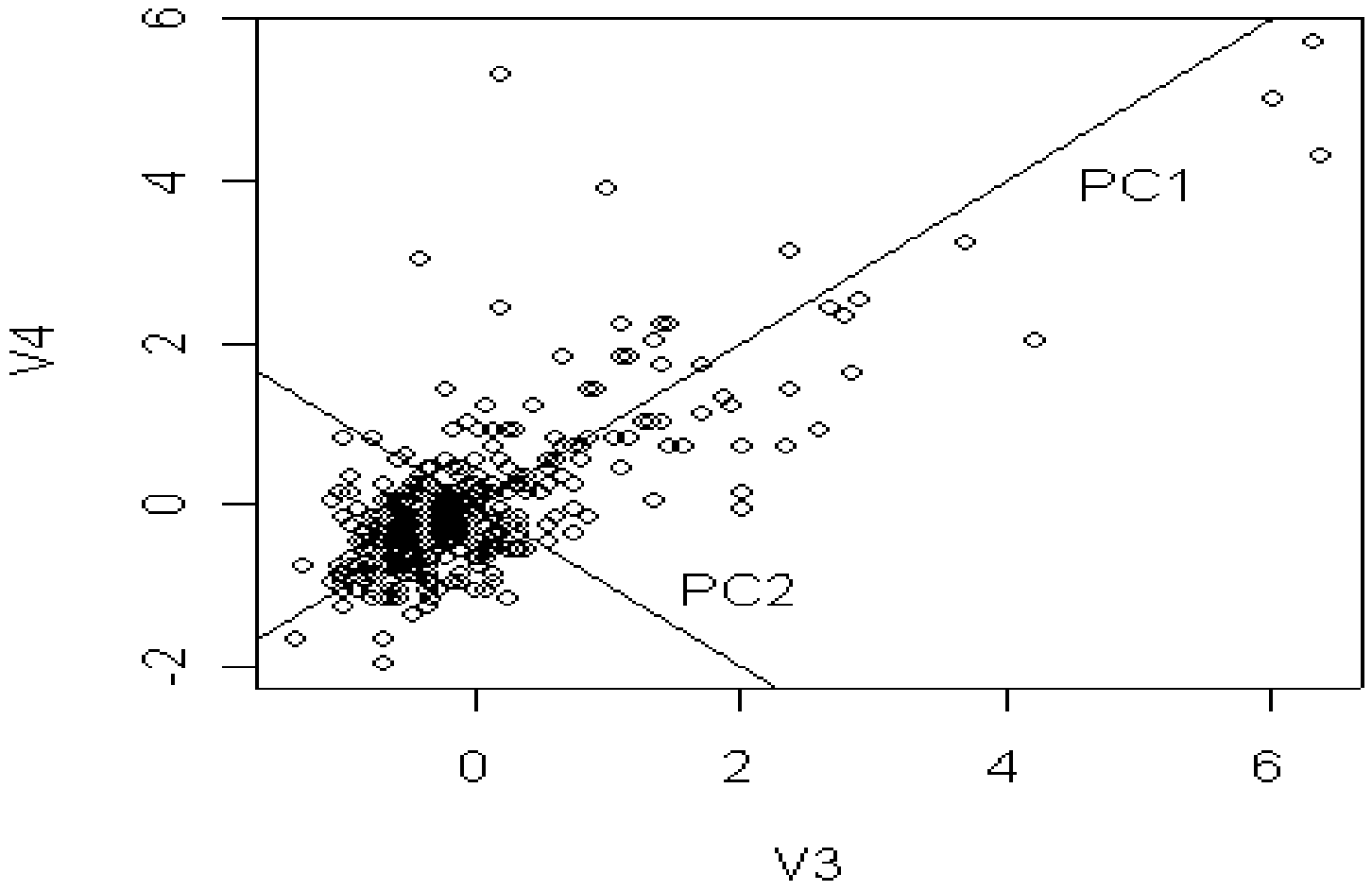
Desde un punto de vista geométrico, aplicar los componentes principales es equivalente a aplicar una rotación de los ejes coordenados.

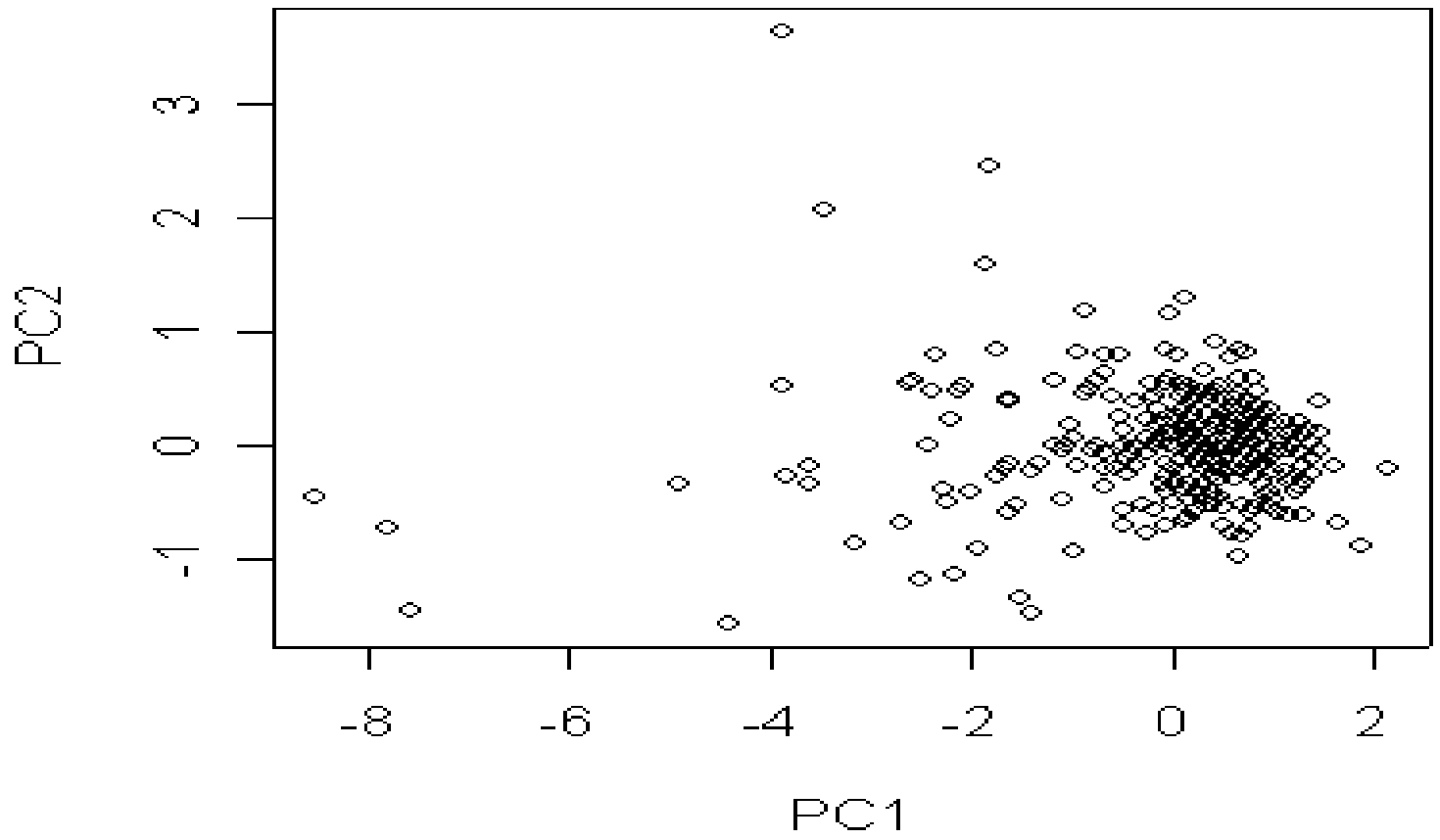
Ejemplo: Bupa (p=q=2)

```
> bupapc=prcomp(bupa[,c(3,4)],scale=T,retx=T)
> print(bupapc)
Desviación estándar:
[1] 1.3189673 0.5102207

Rotation:
         PC1        PC2
V3 -0.7071068 -0.7071068
V4 -0.7071068  0.7071068
```

effect of PCA





Notar que PC1 y PC2 no están correlacionadas

Hallando los Componentes Principales

Para determinar los Componentes Principales Z, debemos encontrar una matriz ortogonal V tal que

i) Z= $\hat{X}$ V ,donde $\hat{X}$ es obtenida de normalizar cada columna de X(cada columna se resta por la media y se divide entre la raíz cuadrada de las desviaciones cuadráticas.

Se puede demostrar que VV'=V'V=I

y ii) Z'Z=($\hat{X}$V)'($\hat{X}$ V) =V'$\hat{X}$' $\hat{X}$ V

=diag(λ_1,….,λ_p)

donde los λ_j's son los valores propios de la matriz de correlación $\hat{X}$' $\hat{X}$. Asi que V se encuentra usando la descomposición de la matriz de correlacion (o de covarianza) en sus valores propios.

La matriz V se llama matriz de cargas y contiene los coeficientes de todas las variables en cada componente principal.

De (ii) el j-ésimo componente principal Z_j tiene una desviación estándar $\sqrt{\lambda_j}$ y puede ser escrito como:

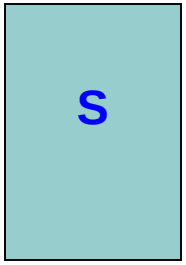
$$Z_j = v_{1j}\hat{X}_1 + v_{2j}\hat{X}_2 + ...+ v_{pj}\hat{X}_p$$

Donde $v_{j1}, v_{j2}, \ldots.. v_{jp}$ son los elementos de la j-ésima columna en V y X_k es la k-esima columna de X .

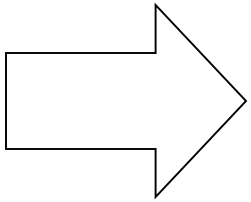
Los valores calculados del componente principal Z_j son llamados los valores rotados o simplemente los "scores".

PCA COMO UN PROBLEMA DE OPTIMIZACIÓN

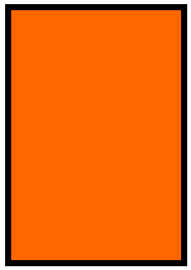
X (nxp)



S=X'X, matriz de corr o de cov



Z(nxp)



Matriz de componentes

$$\mathbf{Z_k} = \text{argmax var } (X\gamma)$$
$$\gamma'\gamma = 1$$

Sujeto a la restricción de ortogonalidad

$$\gamma_j' \, \mathbf{S} \, \gamma_k = 0 \qquad \forall \quad 1 \leq j < k$$

Selección del número de componentes principales

En la practica solo se usa un numero pequeño de componentes principales. Existen muchas alternativas (Ferre, 1995), pero las más usadas son:

- Escoger el número de componentes con una proporción acumulada de valores propios (i.e, varianza) de por lo menos 75 por ciento.
- Escoger hasta el componente cuyo valor propio es mayor que 1. Usar "Scree Plot".

Ejemplo: Bupa

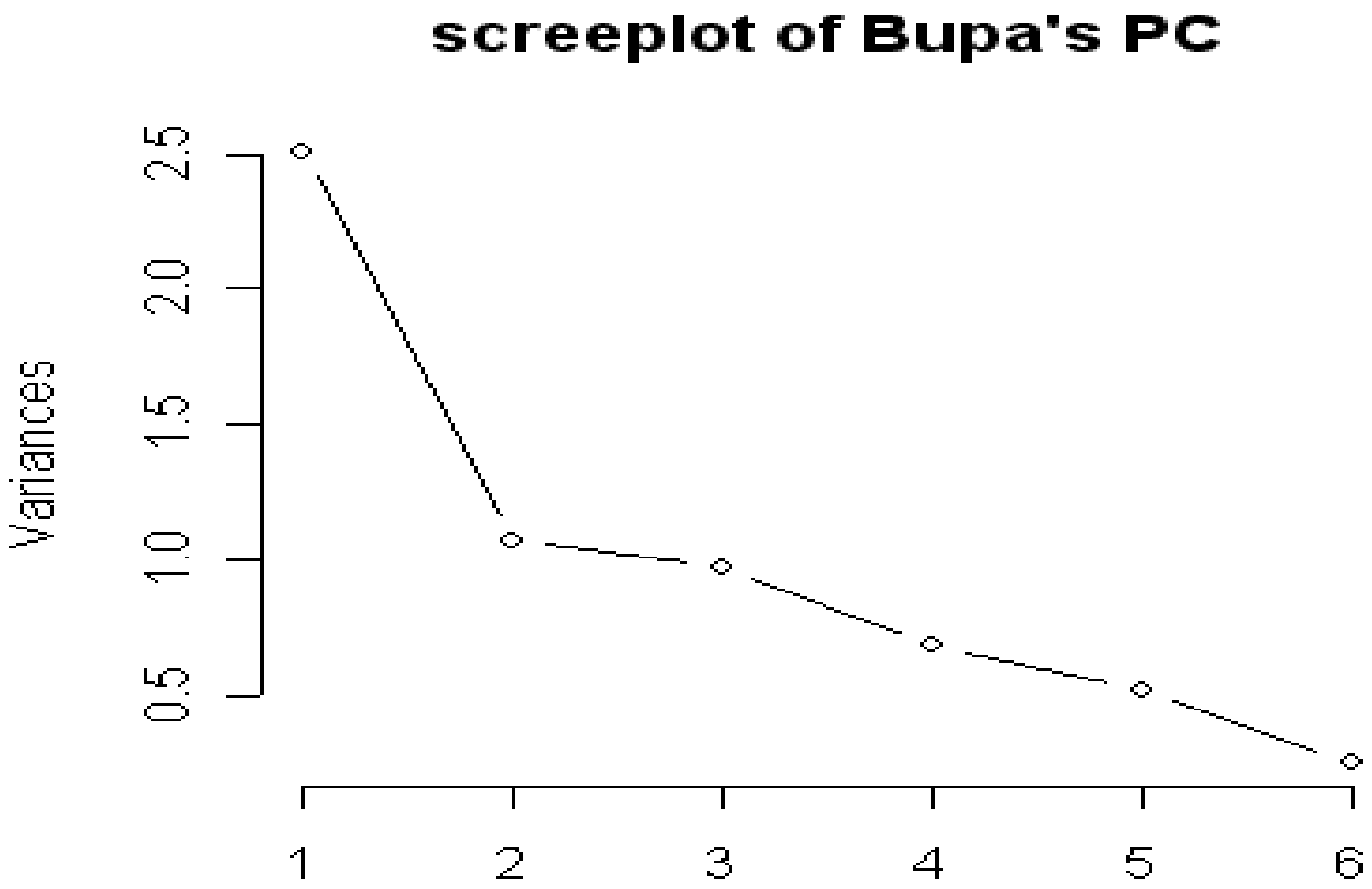
```
> a=prcomp(bupa[,-7],scale=T)
> print(a)
Desviación estándar:
[1] 1.5819918 1.0355225 0.9854934 0.8268822 0.7187226 0.5034896

Rotation:
        PC1         PC2         PC3         PC4         PC5          PC6
V1 0.2660076  0.67908900  0.17178567 -0.6619343  0.01440487
    0.014254815
V2 0.1523198  0.07160045 -0.97609467 -0.1180965 -0.03508447
    0.061102720
V3 0.5092169 -0.38370076  0.12276631 -0.1487163 -0.29177970
    0.686402469
V4 0.5352429 -0.29688378  0.03978484 -0.1013274 -0.30464653 -
    0.721606152
V5 0.4900701 -0.05236669  0.02183660  0.1675108  0.85354943
    0.002380586
V6 0.3465300  0.54369383  0.02444679  0.6981780 -0.30343047
    0.064759576
```

Ejemplo (cont)

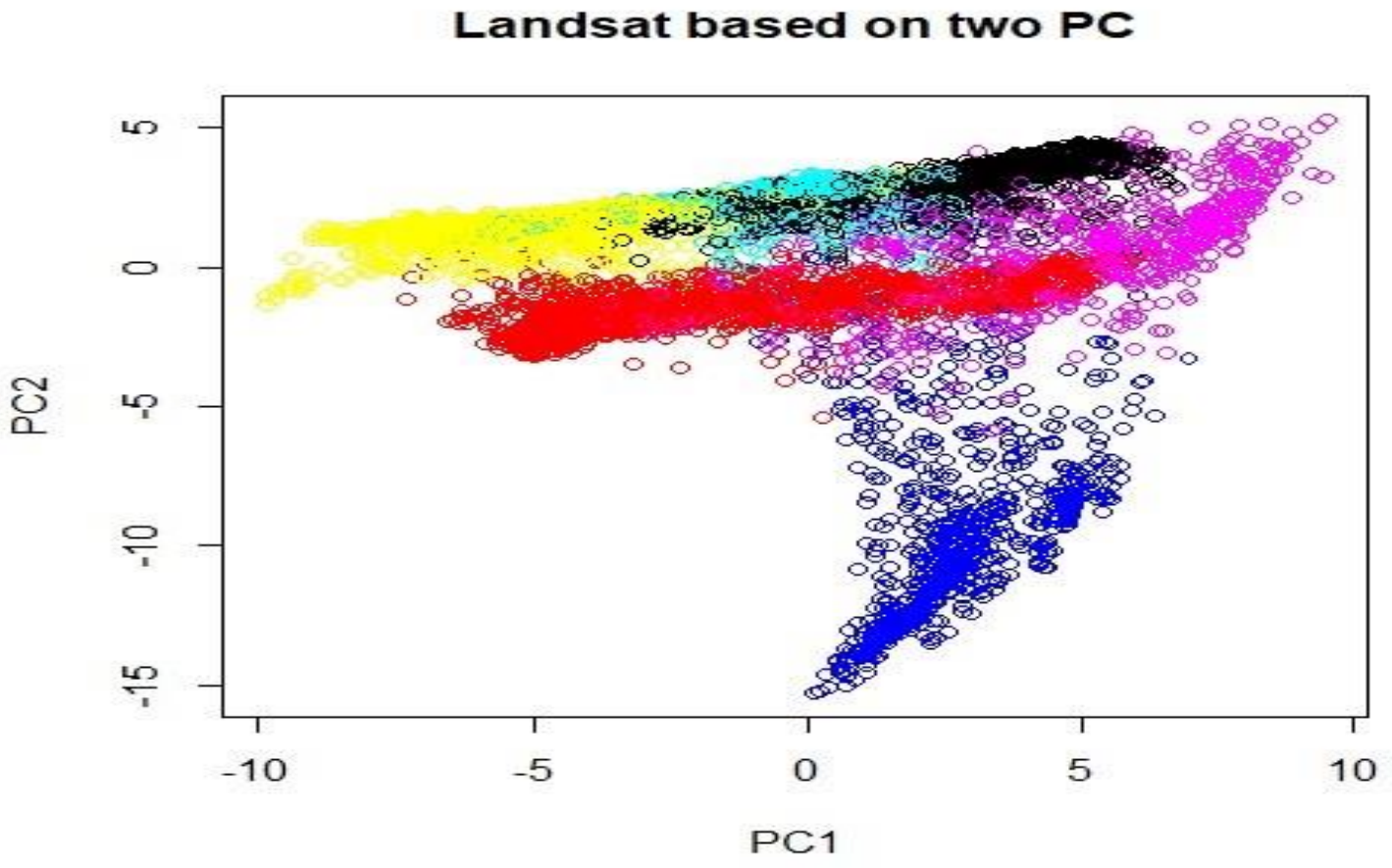
```
> summary(a)
Importancia de componentes:
                          PC1   PC2   PC3   PC4    PC5    PC6
Desviación estándar     1.582 1.036 0.985 0.827 0.7187 0.5035
Proportion of Variance 0.417 0.179 0.162 0.114 0.0861 0.0423
Cumulative Proportion  0.417 0.596 0.758 0.872 0.9577 1.0000
>
```

- Se podrían usara dos o tres componentes para predecir/



El screen-plot recomienda elegir dos componentes principales

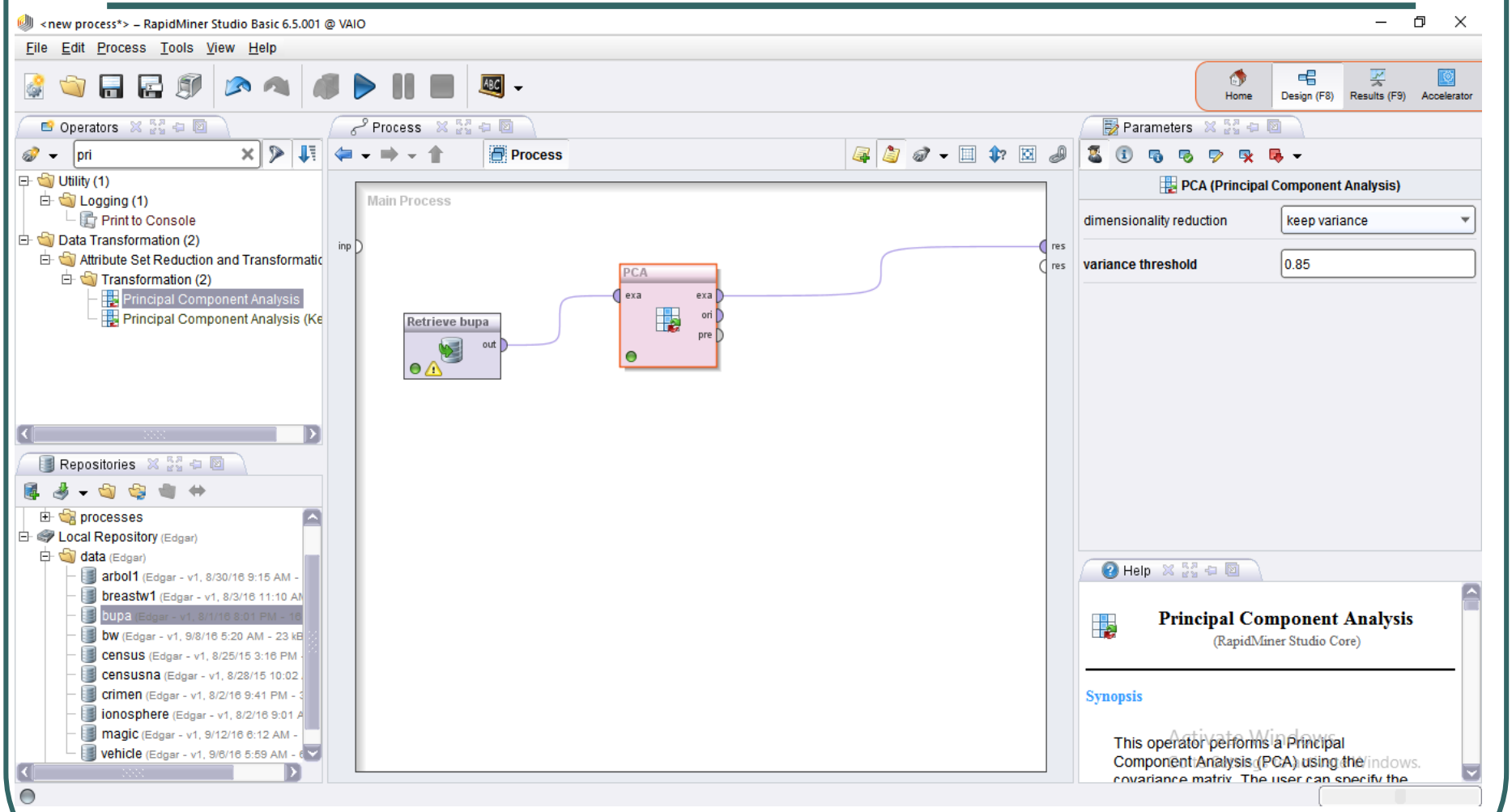
Usando PCA para visualizar datos de alta dimension



PCA en Python

La Libreria scikit-learn hace PCA
From sklearn.decomposition import PCA

PCA in Rapidminer



Comentarios

- Muchos estudios han demostrado que PCA no da buenas predicciones en clasificación supervisada.
- Mejores alternativas: PLS (partial least squares) generalizados (Vega,2004) y PCA supervisados (Hastie, Tibshirani, 2004, Acuna and Porras, 2006).
- Para mejorar el rendimiento de componentes principales es mejor hacer primero selección de variables.