laSalle
UNIVERSITAT RAMON LLULL

Master en BIG DATA

ESTADÍSTICA

Clustering no Supervisado y Supervisado

Práctica 3

La Salle, Universitat Ramon Llull

Autores:
Jaume Feliubadaló Rubio
Juan José Carbonell Moral
Miguel Ángel Tarrason Moron

Profesor:
Jordi Cortés Martínez

Fecha de entrega:
23/12/2018

Índex

Índex de figures

1 Objetivo

El análisis de clusters, también nombrado como segmentación de datos, tiene una variedad de objetivos. Dichos objetivos estan relacionados con agrupar o segmentar un set de datos en clusters, de modo que los que están dentro de cada agrupación están más estrechamente relacionados entre sí que los objetos asignados a diferentes agrupaciones. El Clustering también se usa para formar estadísticas descriptivas para determinar si los datos consisten o no en un conjunto de subgrupos distintos, cada grupo representa objetos con propiedades sustancialmente diferentes.

Disponemos de un set de datos, el cual contiene información sobre movilidad recogida en teléfonos móviles. Se pide clasificar la actividad de los usuarios en 6 niveles: tumbado (laying), sentado (sitting), de pie (standing), caminando (walk),bajando escaleras (walkdown) o subiendo (walkup).

Se divide el análisis en dos partes, en la primera se agruparan las respuestas de los sensores usando el método de clustering no supervisado (K-means) y en la segunda se construye un modelo predictivo capaz de clasificar los individuos en cada instante utilizando la técnica de Clustering supervisado (algoritmos tales como: KNN, Naive Bayes, Conditional trees, Random Forests y SVM).

2 Clustering no supervisado [K-means]

El método de clustering K-means es usado para encontrar clusters y sus respectivos centros en un set de datos sin la variable respuesta. El algoritmo K-means es uno de los métodos de agrupamiento (en clusters) de descenso iterativo más populares. Tiene inconvenientes de los cuales destacamos que se basa en distancias medias, eso repercute en que los outliers afectan al resultado.

$$d(x_i, x_{i'}) = \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 = ||x_i - x_{i'}||^2$$

Figura 1: Algoritmo K-means.

2.1 Primer análisis

Se realiza un primer análisis de los datos, con el cual se extrae información tal como: inercia total, entre grupos e intra. Del cociente entre la inercia entre y la total se obtiene la variabilidad explicada.

Inercia Total	$\sum_{k=1}^{K} \sum_{q=1}^{Q} \sum_{i=1}^{I_q} (y_{iqk} - \bar{y}_k)^2$	*[Distancia de cada punto al centro de gravedad]*
Inercia Entre-grupo	$\sum_{k=1}^{K} \sum_{q=1}^{Q} I_q (\bar{y}_{qk} - \bar{y}_k)^2$	*[Distancia del centro de los grupos al centro global]*
Inercia Intra-grupo	$\sum_{k=1}^{K} \sum_{q=1}^{Q} \sum_{i=1}^{I_q} (y_{iqk} - \bar{y}_{qk})^2$	*[Distancia de cada punto al centro de su grupo]*

Figura 2: Inercias K-means.

Destacar que este último parámetro mencionado es el objetivo a maximizar, dicho parámetro es un indicador de la calidad de una partición. La variabilidad explicada para un número de clusters de 6, tiene un valor de 0,703997.

$$Variabilidad\ explicada = \frac{Inercia \quad entre - grupos}{Inercia\ total}$$

Figura 3: Variabilidad explicada.

2.2 Análisis Regla del codo y NbClust

En primera instancia se escoge un número de clusters igual a 6, es con la denominada regla del codo con la que se valora qué número de clusters se debe escoger.

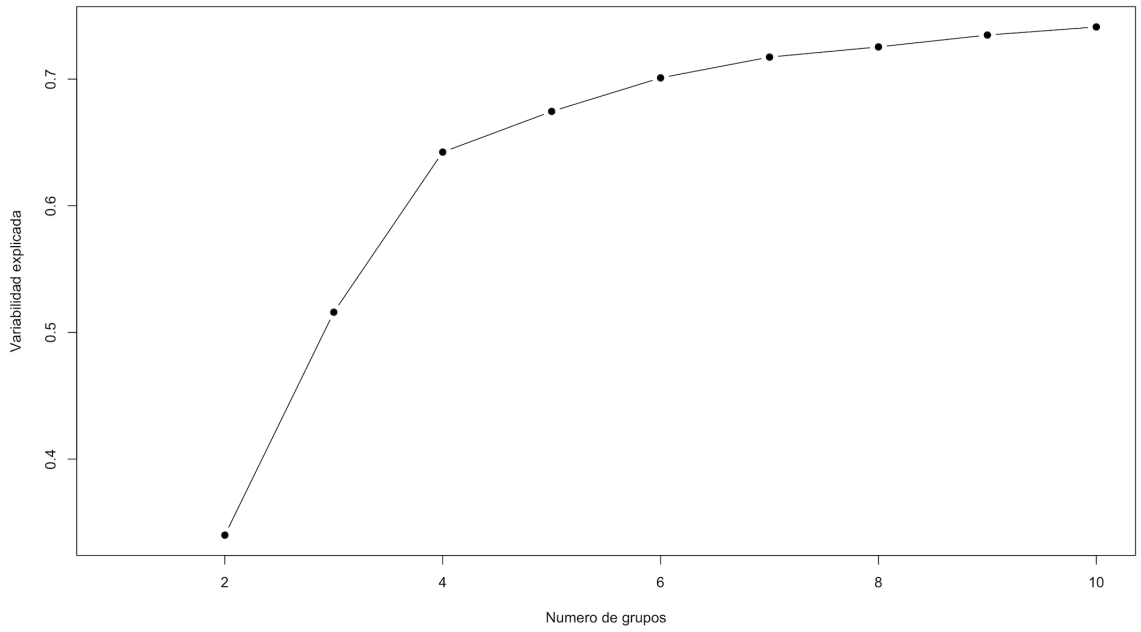

Figura 4: Regla del codo.

Se aplica la función Nbclust con el objetivo de determinar el número óptimo de clusters, si bien la regla del codo es una determinación del número de clusters visual, en este caso se trata de un resultado concreto y sin interpretación alguna (en la regla del codo no se determina al 100% que número de clusters se debe escoger).

2.3 Resultado y conclusión

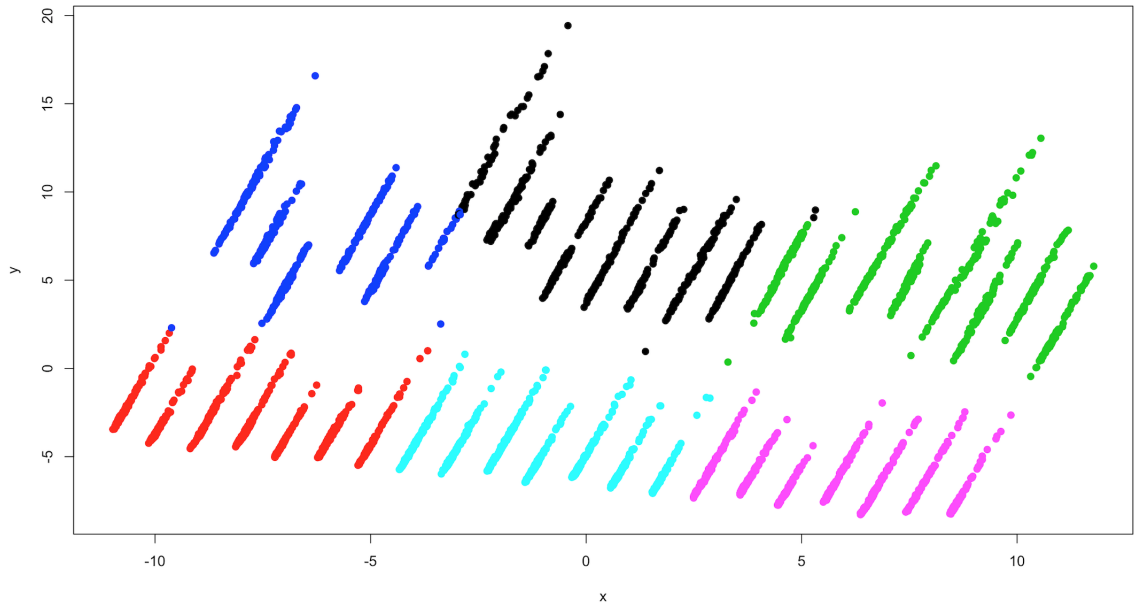

Figura 5: Plot de los 6 clusters resultantes.

En la Figura 5, se aprecia el resultado de aplicar el algoritmo a la muestra de datos. Se aprecian claramente 6 clusters, cada uno asignado a un color diferente. Resulta evidente la clasificación según la actividad, se podría hablar de que los clusters con color rosa, azul celesta y rojo representan las actividades de laying, sitting y standing mientras que los 3 restantes representan a el resto de actividades (actividades que representan movimiento).

Se escogen 6 clusters en base a que es el número de clusters que da mayor valor de variabilidad explicada, se obvia el resultado del randIndex.

3 Clustering Supervisado

3.1 Naive Bayes

Este método de clasificación se basa en el teorema de Bayes el cual dice que sean A y B dos sucesos no nulos, se cumple:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Figura 6: Teorema de Naive Bayes.

Para el uso del algoritmo de Naive Bayes se requiere tener presente la premisa de independencia de las variables. Dicha premisa implica que cada 'feature' influya de forma autónoma en determinar la clase. Naive Bayes también requiere que cada probabilidad condicional sea distinta a cero (sucesos no nulos).

3.1.1 Validación independencia

Para la validación de la premisa independencia se debe retirar la variable identificativa del individuo y la respuesta. Una vez retirada y ejecutado el código correspondiente se observa en la figura 3 que las variables se mueven en unas correlaciones altas, se concluye que la premisa no se cumple.

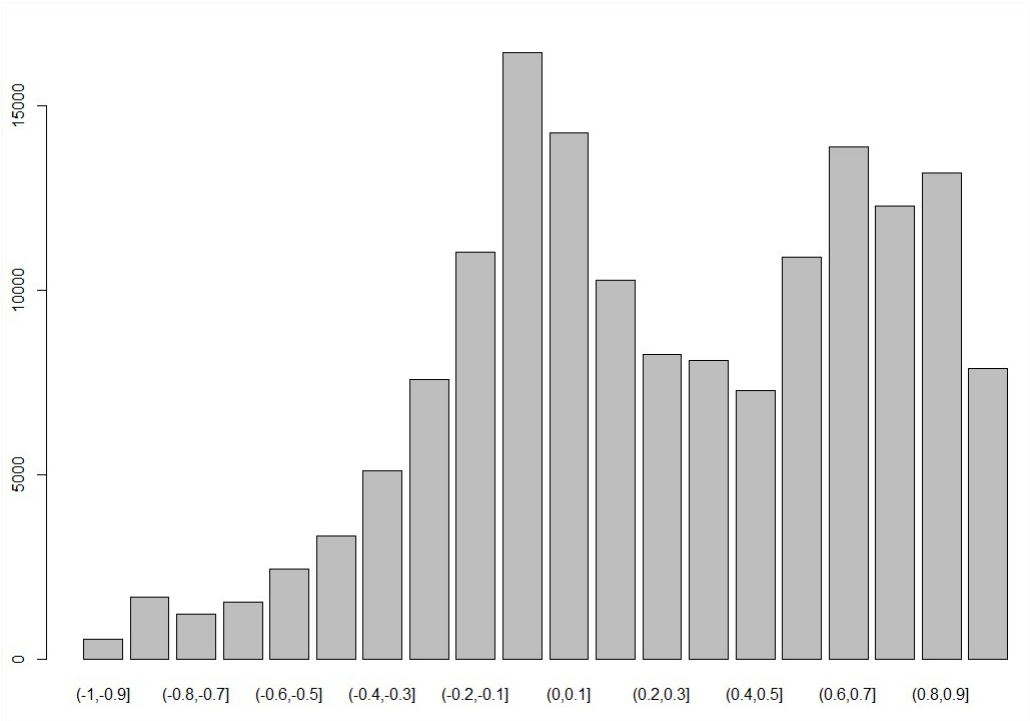

Figura 7: Validación premisa independencia.

3.1.2 Capacidad predictiva

Se aplica una división de los datos de entrenamiento, dichos datos se dividen con el fin de realizar una predicción. La división corresponde a 2/3 los datos de entrenamiento y 1/3 los datos de

test (ambos con origen en los datos de entrenamiento con 5000 observaciones). Otra vez más, se debe suprimir la variable identificativa de los individuos.

Observamos que la distribución que realiza el algortimo es correcta, no agrupa más individuos en un estado que en otro.

laying	sitting	standing	walk	walkdown	walkup
700	598	641	575	455	523

Figura 8: Agrupación por clases.

Cuando visualizamos la capacidad predictiva sobre el conjunto de entrenamiento observamos que las clases que mejor clasifica son laying y standing, con dicha clasificación se obtiene un 80 por ciento de capacidad predictiva.

preds	laying	sitting	standing	walk	walkdown	walkup
laying	287	3	0	0	0	0
sitting	9	138	9	0	0	0
standing	0	123	261	0	0	0
walk	0	0	0	160	8	8
walkdown	0	0	0	34	161	21
walkup	1	2	4	41	35	203

Figura 9: Capacidad predictiva según clases.

3.1.3 Optimización algoritmo

Se realiza una revisión visual de la probabilidad que tiene cada individuo de pertenecer a una clase, de esta manera se entiende la clasificación por parte del algoritmo en cuestión. En dicha revisión, se percibe que hay individuos que los clasifica muy bien, pero, por ejemplo, en el individuo 16 el algoritmo duda, ya que pinta laying y sitting. Esta observación nos indica que hay que mejorar.

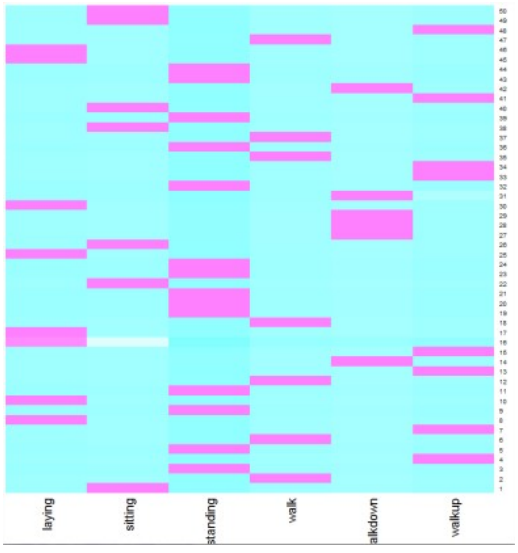

Figura 10: Revisión visual.

La probabilidad que le asigna a cada variable de acierto es muy elevada, la cual se encuentra por encima del 50% en la gran mayoría de ellas. Con las validaciones visuales se observa que se requiere de realizar una discriminación de variables, es decir, eliminar aquellas muy correlacionadas entre sí. Para ello nos ayudamos del índice Gini, se realiza validación gráfica y se observa que casi todas tienen un porcentaje alto de pertenecer a más de dos respuestas (clases).

Se intenta esclarecer cuales de las variables son más predictoras, obtenemos 9 variables de las cuales 6 quedan clasificadas dentro de 3 respuestas diferentes. Con estos datos y validaciones visuales nos vemos obligados a filtrar las variables con más correlación; de 561 variables nos quedamos con 132.

Aplicamos el algortimo de Naive Bayes nuevamente, con la premisa de la independencia ya cumplida. Se obtiene un 72%, entendemos que teniendo en cuenta que se ha despreciado 429 variables es un buen dato de capacidad predictiva.

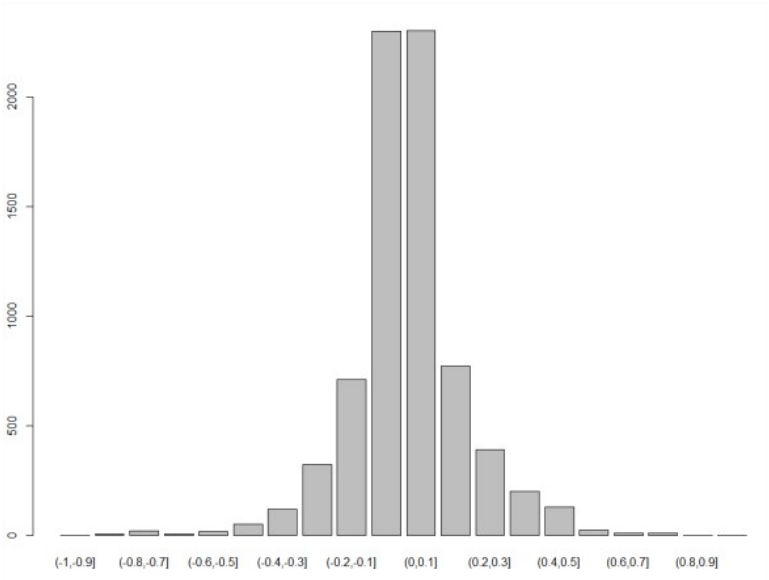

Figura 11: Premisa independencia 2.0.

En este conjunto de datos realizar Bagging o Boosting no mejoraría significativamente, ya que, como hemos indicado, las variables al no cumplir la premisa independencia, es necesario retirar la gran mayoría.

3.2 Conditional Trees

A la hora de aplicar este algoritmo no debemos tener presente la variable respuesta. Las variables predictoras pueden ser del tipo numéricas y categóricas. En este caso que nos ocupa se cumple que tenemos variables numéricas como predictoras y variable categórica de estado en la respuesta. Representar árboles de mucha profundidad resulta de un grado de complejidad alto. Se realiza una selección de un 70% sobre los datos de entrenamiento y eliminamos la variable identificativa de cada individuo. Realizamos una primera inspección visual con un árbol de profundidad de 3.

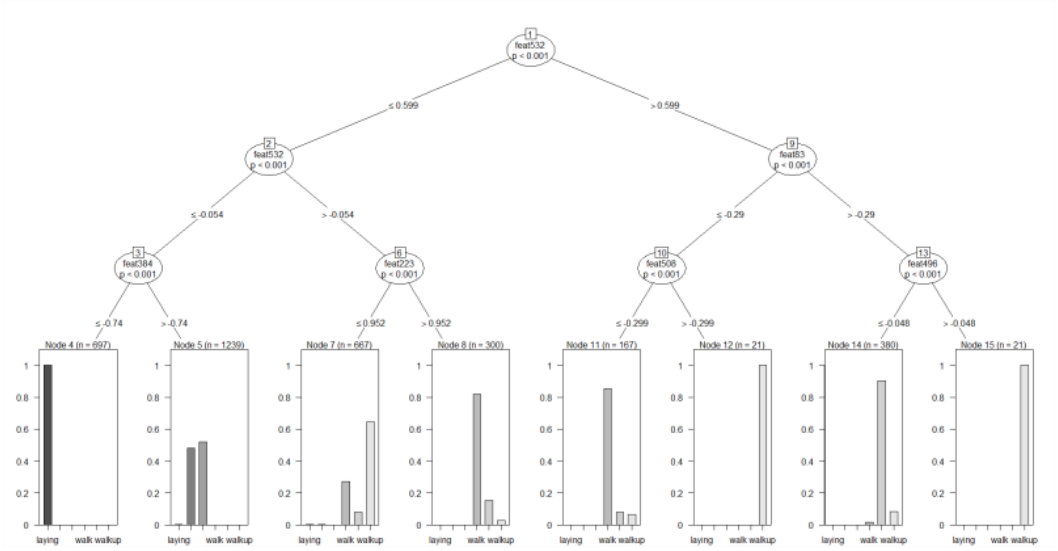

Figura 12: Visualización conditional trees [Profundidad = 3].

El algoritmo, teniendo presente todas las variables, identifica la característica 532 y el valor 0,59 como punto de inflexión para comenzar a clasificar (nodo raíz). Con esto, clasifica el nodo hoja más a la derecha, con una probabilidad de más de 0,9 va a pertenecer a subir escaleras y clasifica 21 individuos. El problema es que hay nodos hojas que pertenecen a demasiados individuos, por ejemplo el nodo 5 tiene 1239 y el último solo 21. Hemos de mejorar la clasificación a nivel de pertenencia y asignación a clases repuesta.

3.2.1 Mejora predictiva; Conditional Trees

Intentamos mejorar la capacidad predictiva sin limitar la profundidad del árbol, con lo que obtenemos una capacidad predictiva del 0,888 89%. Intentamos mejorar la predicción sobre el total de datos de entrenamiento, obviamos los diferentes versiones como Bonferroni, Montecarlo etc, entendemos que el algoritmo por defecto es el óptimo. Con esto optamos por personalizar el algoritmo con una preselección de variables igual a 300 de un máximo de 561, la profundidad de los árboles se verá limitada a que el p-valor para realizar una división ha de ser de 0,95 y queremos que para realizar la división hayan 20 observaciones como mínimo. Con estas premisas, se ha conseguido un 90% de probabilidad de acierto sobre el total de datos de entrenamiento. Con configuración de peso en nodo a 20, p-valor 0,90 se ha conseguido un 0,89 en datos Test.

```
pred       laying sitting standing walk walkdown walkup
  laying      296       0        0    0        0      0
  sitting       1     236       40    0        0      0
  standing      0      30      233    0        0      0
  walk          0       0        1  206       16     23
  walkdown      0       0        0   17      179     19
  walkup        0       0        0   12        9    190
> sum(diag(t))/sum(t)
[1] 0.8885942
```

Figura 13: Ejemplo matriz de confusión y predicción de acierto.

3.3 Random Forest

El método de clustering Random Forest se basa en la generación de muchos árboles condicionales, en esta práctica se hace en base a la selección aleatoria de las variables, no se valora realizar

Bootstrap ya que requiere un coste computacional importante. En este algoritmo las variables pueden ser categóricas o numéricas, tanto en predictoras como en la respuesta.

Como no necesita datos de entrenamiento, se decide trabajar sin realizar subdivisión, eliminando la variable identificativa del individuo.

Se realiza una primera estimación con 50 árboles, indicando que tenga presente la importancia de las variables, se obtiene:

```
ntree      OOB      1      2      3      4      5      6
   50:     3.24%    0.00%  6.71%  6.12%  1.98%  3.19%  1.46%
```

Figura 14: Estimación 50 árboles.

Para cada clase da el error de predicción y el error de predicción global (00B), la clase 1 da sobre un 0%, tiene excelente predicción, la clase 2 y 3 da un 6%, predice muy bien. Para el resto la tasa de error son muy bajas entorno del 1% al 3%, en global, acertamos un 96,6% ¡- 100 - columna 00B. Dicha predicción se puede mejorar, generar esta probabilidad ha sido fácil (computacionalmente hablando), pero, vamos a ver graficamente si estamos sobre una probabilidad óptima:

Se observa en la Figura 11 que las variables tienen oscilaciones, excepto la roja discontinua, la clase 1 (laying), la tasa de error (linea negra) oscila bastante, en conclusión, el modelo requiera aun mucho entrenamiento con los datos.

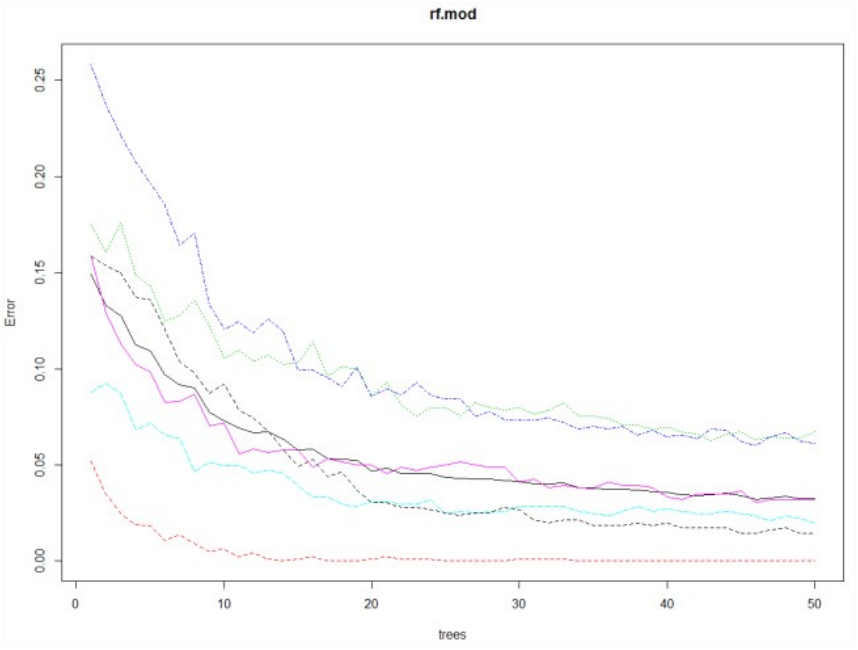

Figura 15: Plot oscilaciones.

Se realiza una validación de las variables más o menos importantes y que pueden ayudar a mejorar el modelo en la toma de decisiones a la hora de dividir y generar nuevas ramas.

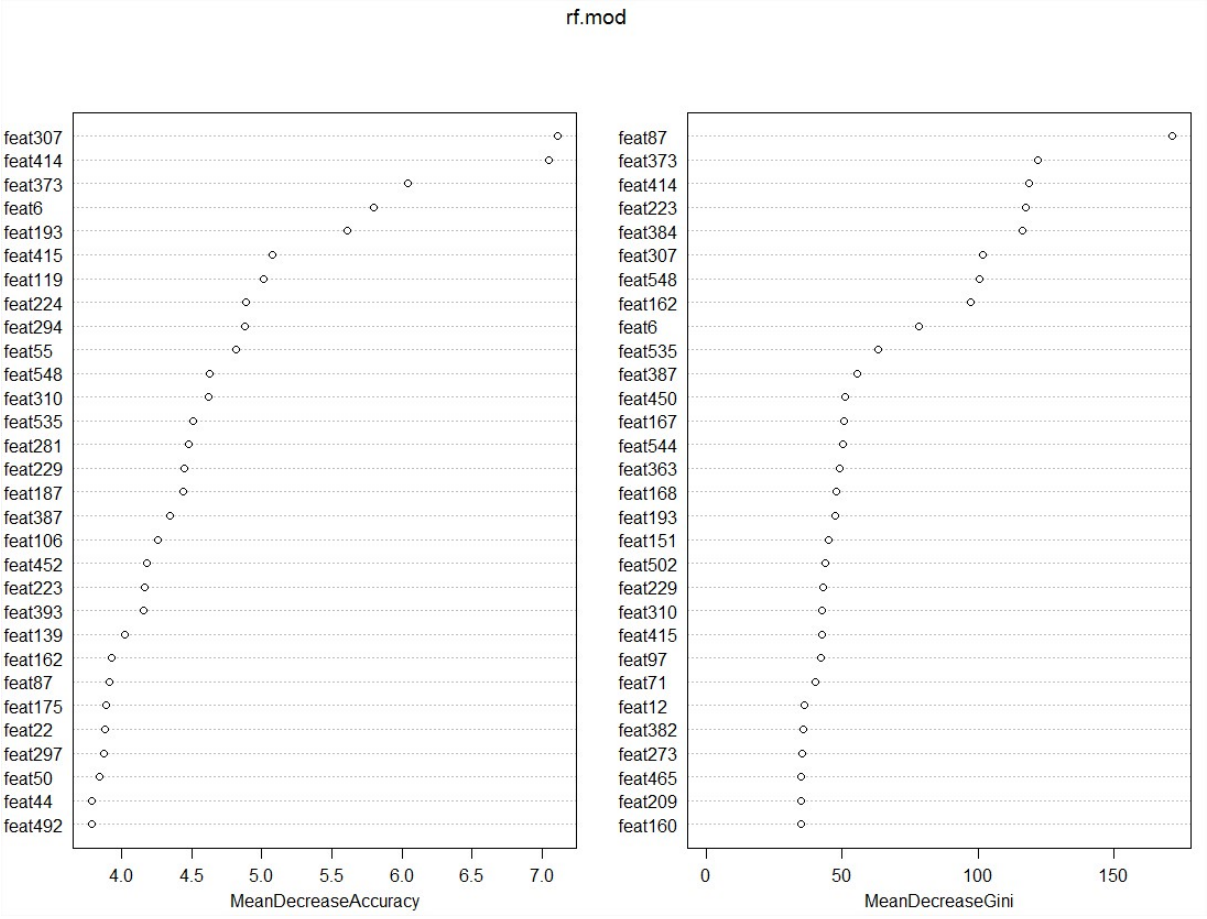

Figura 16: Importancia variables.

Observamos que en los dos medidores, entre las tres primeras están 414 y 373, a priori seleccionamos el indicador MeanDecreaseAccuracy. Nos interesa que estas variables tengan más peso en la decisión del algoritmo.

Intentamos definir de estas variables cuantas queremos que sean escogidas como mucho, realizamos el cálculo:

```
mtry = 23  OOB error = 3.2%
Searching left ...
mtry = 12       OOB error = 3.36%
-0.05 0.05
Searching right ...
mtry = 46       OOB error = 3.48%
-0.0875 0.05
```

Figura 17: Tunear parámetro mtry.

Obtenemos que con 23 nos ofrece la tasa de error menor. Sólo nos queda intentar mejorar el algoritmo en base a estos datos y el número de árboles de entrenamiento que queramos fijar. Decidimos fijar la importancia de las variables, número de árboles a 2000 y mtry a 10, con estos datos hemos conseguido la mejor tasa de error.

Como hemos trabajado sobre los datos de entrenamiento en total, la probabilidad que nos ha dado contra los datos de test ha sido de 97,85 en Shiny. Realizando una validación grafica del modelo observamos que ahora sí, representa las clases totalmente relajadas.

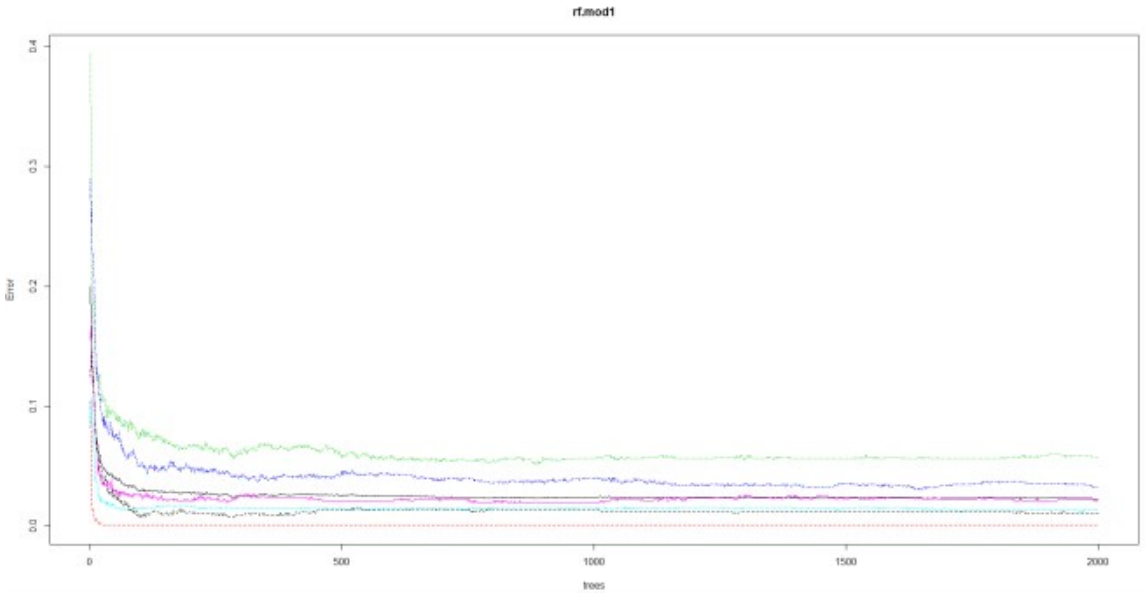

Figura 18: Validación gráfica.

Esto indica que a mayor número de árboles de entrenamiento, se consiguen mejores tasas de error, pero llega un momento que la mejora son décimas y el coste computacional aumenta considerablemente.óptima

3.4 SVM: suport vector machine

El SVM es un algoritmo basado en la partición binaria de carácter no probabilístico. Se trata de hallar un hiperplano que separe de forma óptima los puntos de cada clase.

L·Las funciones Kernel logran que el coste de calcular estos productos escalares sea similar al del espacio original."

Entre sus ventajas cabe destacar que es un problema de optimización que siempre converge. Se trata de que la distancia de los puntos al hiperplano sea la mínima posible.

$$d_i = \frac{1}{\|\beta\|_2} \cdot y_i \cdot f(x_i)$$

β: coeficientes del hiperplano
y_i: clase de la observación *i (-1 o 1)*
$f(x_i)$: Clasificación del punto en función del hiperplano:

$$f(x_i) = \beta_0 + \beta' \cdot x_i$$

Figura 19: Distancia de cualuier punto al hiperplano.

Al ejecutar por pimera vez el código, sin tunear ningún parametro da una predicción de 98.46%.

```
pr          laying sitting standing walk walkdown walkup
  laying       997       1        0    0        0      0
  sitting        0     829       42    0        0      0
  standing       0      34      873    0        0      0
  walk           0       0        0  810        0      0
  walkdown       0       0        0    0      659      0
  walkup         0       0        0    0        0    755
> sum(diag(t))/sum(t)
[1] 0.9846
```

Figura 20: Resultado SVM: 98.46%.

Se aprecia claramente que el algoritmo a escoger será el SVM. Su clasificación es más precisa que los algoritmos anteriormente mencionados en el presente trabajo.

Procederemos a realizar alguna mejora al algoritmo, tal como optimizar la función kernel y escoger el 'cost' es decir la distancia al hiperplano que mejor se ajuste, con el objetivo de aumentar el tanto por ciento de predicción.

Una vez ejecutada la linia de código correspondiente (función tune), se obtiene el siguiente resultado:

```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost
  0.1

- best performance: 0.0194

- Detailed performance results:
   cost  error  dispersion
1 1e-02 0.0220 0.006992059
2 2e-01 0.0208 0.006408328
3 1e-01 0.0194 0.005420127
4 1e+00 0.0208 0.006746192
5 5e+00 0.0218 0.006957011
6 1e+01 0.0218 0.006957011
7 1e+02 0.0218 0.006957011
```

Figura 21: Tune de parámetros SVM.

Se indica cual ha sido la mejor 'perfomance' y el mejor parámetro de 'cost', en nuestro caso resulta ser de 0.1.

En shiny se consigue una predicción de 98.45%. Es pues, el SVM, el algoritmo a escoger para la clasificación de las distintas actividades de los individuos.

Comentar que no se valora la opción del algoritmo de KNN debido a que es un algoritmo que se debe usar para 20 variables, en nuestro caso se tienen más de 550 variables y cremos que no vale la pena invertir tiempo en dicho algoritmo sabiendo de antemano dicha información.