Universitat Oberta de Catalunya

uoc.edu

05.583 Aprenentatge Computacional

PRÀCTICA 2017-2018

Luis Enrique Arribas Zapater 17745245D

EJERCICIO 1

Descripción en detalle del algoritmo RkNN.

Justificación

Partiendo de los algoritmos k Nearest Neighbor y Random Forest (múltiples árboles de decisión) el algoritmo RkNN toma determinados conceptos de ambos modelos de clasificación para "hibridarlos" en un único algoritmo descrito extensamente en el artículo *"Random KNN feature selection - a fast and stable alternative to Random Forests. BMC Bioinformatics. 2011;12:450. doi: 10.1186/1471-2105-12-450"* de Li, Harner y Adjerh.

Determinados conjuntos de datos con una cantidad *p* de características, variables o atributos (en adelante se utilizarán indistintamente) mucho mayor a la cantidad *n* de elementosdel conjunto de datos (p>>n), se vuelven intratables y los métodos de reducción de dimensionalidad como LDA no pueden aplicarse de forma efectiva.

RkNN es un algoritmo de clasificación que reduce la dimensionalidad a las variables más relevantes, consiguiendo precisión y coste aceptables.

En el artículo la necesidad de reducción de características nace de la existencia de más de 10.000 genes tomados como variables para la predicción de determinadas enfermedades congénitas.

Decision tree, Random Forest y kNN

Veamos a continución los algoritmos sobre los que se fundamenta RkNN.

Mediante algoritmos de selección interna de variables (Decision trees y Random forests) puede abordarse el problema, pero con inconvenientes en cuanto a la precisión se refiere. Decision tree es un algoritmo basado en la clasificación por nodos (donde cada nodo corresponde a una variable) y la determinación de un umbral en cada nodo, de forma que si el valor de la variable del individuo que se está clasificando supera o no dicho umbral, se clasifica al individuo en una categoría o por contra se avanza hasta otro nodo donde se compara otra variable. Este proceso se reproduce hasta recorrer el árbol hasta un nodo hoja.

Los árboles de decisión se vuelven inestables debido a sus estructuras jerárquicas, en las que la presencia de ruido puede propagar errores de clasificación a través del arbol. Un error cerca de la raiz puede definir un árbol completamente distinto con la consiguiente pérdida de precisión.

Random forest es una versión mejorada del algoritmo anterior en la que se genera una colección de árboles cuyas variables son seleccionadas aletoriamente. Se realiza una clasificación en cada uno de los árboles del bosque y en la etapa final se produce una votación entre los distintos resultados para decidir la clase del individuo. De esta forma puede mitigarse la acción del ruido.

Por otro lado el algoritmo kNN es un método de clasificación no paramétrico. Es simple en cuanto a implementación y es estable.

K-means selecciona dos o más elementos del conjunto de datos donde fija los centroides de los respectivos cluster. Así para *n* clusters se seleccionarán *n* centroides.

Para cada elemento *e* se calcula la distancia a cada uno de los *n* centroides. *e* es categorizado en el cluster a menor distancia y se recalcula el centroide del cluster teniendo en cuenta el nuevo componente para el centro de masas común.

De esta forma los *n* centroides se desplazan en cada iteración y debe por tanto recalcularse para todos lo elementos la pertenencia a cada cluster. Para grandes volúmenes de datos esto puede suponer problemas de computación.

Cuando se alcanza un estado en el que los centroides no se desplazan al iterarse, se alcanza una situación “estable”. Se ha alcanzado la convergencia y el algoritmo finaliza.

Construcción del modelo en base a la selección de características por RkNN.

Vistos los conceptos anteriores pasamos a describir el funcionamiento de RkNN como clasificador.

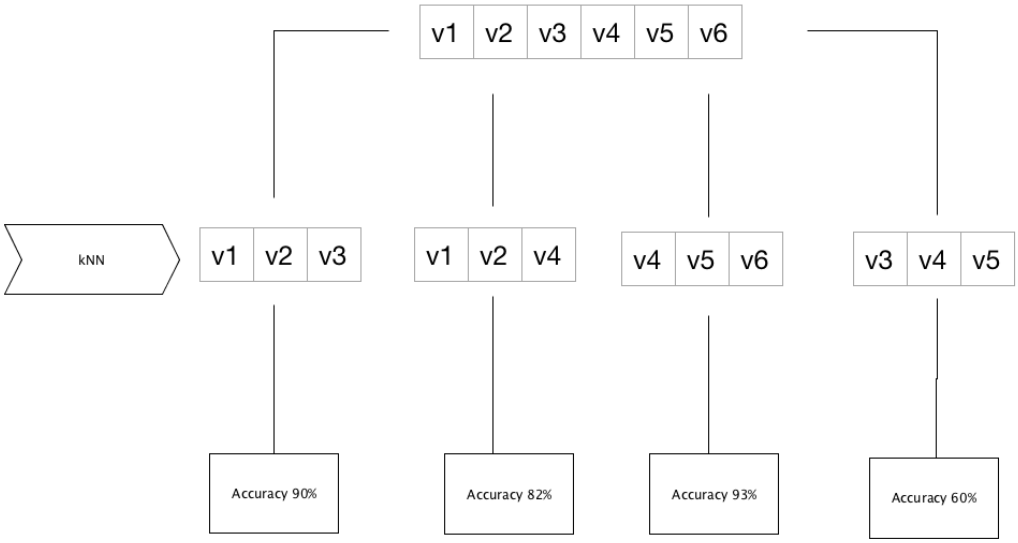
En primer lugar RkNN realiza un ranking de las variables o característica en función de su precisión con un conjunto de datos de entrenamiento.

A continuación de forma iterativa reduce la dimensiionalidad del problema descartando iterativamente las características menos relevantes

Ranking de variables clasificadoras.

Se crean aleatoriamente *i* conjuntos de *j* variables. Cada variable debe aparecer en varios de los *i* conjuntos que usamos para construir los clasificadores kNN.

Introducimos el conjunto de entrenamiento en los calsificadores y obtenemos una precisión para cada conjunto de *j* variables de cada clasificador kNN. La precisión media de los kNN donde



interviene dicha variable es denominada *apoyo de la característica* o *feature support*. Así podemos construir un ranking de relevancia de las variables en base a la precisión media de los clasificadores donde han intervenido dichas variables.

Dicho de otra manera, cada variable participa aleatoriamente en un conjunto de kNNs y en base a los resultados de precisión de dicho conjunto las variables usadas se calsifican como mejores o peores.

Random Set	Accuracy
1	90
2	82
3	93
4	60

Random Set	v1	v2	v3	v4	v5	v6
1	90	90	90			
2	82	82		82		
3			93	93	93	
4				60	60	60
Avg	86,00	86,00	91,50	78,33	76,50	60,00

Ranking	Support
v3	91,50
v1	86,00
v2	86,00
v4	78,33
v5	76,50
v6	60,00

En el ejemplo anterior vemos las particiones del conjunto de variables para su clasificación en el ranking en base a su precisión en la figura de la página anterior. A continuación los cálculos para la elaboración del ranking y finalmente el ranking de relevancia de las variables.

Reducción de variables clasificadoras. Modelo final.

Una vez clasificadas las variables por su relevancia el siguiente paso es determinar cuantas variables son necesarias para la construcción final del modelo.

Iterativamente se ejecuta kNN sobre el conjunto de datos y en cada iteración se reduce la cantidad de variables, siendo descartadas primero las de menor relevancia.

La cantidad de variables eliminadas en cada iteración dependerá del diseño (optaremos por reducir una variable por iteración en esta explicación dado el pequeño tamaño del ejemplo). En el artículo los autores trabajan con un conjunto de datos de más de 10.000 variables, por lo que la reducción se hace en dos fases: una primera reducción geométrica, reduciendo a la mitad el tamaño de *p* en cada iteración y una segunda parte lineal, en la que el tamaño se reduce en 1.

Tras cada ejecución de kNN se calculan las precisiones promedio contra el número de variables. La variebles presentes en la iteración de máxima precisión serán las que conservaremos para el modelo.

Ranking	Support
v3	91,50
v1	86,00
v2	86,00
v4	78,33
v5	76,50
v6	60,00

Continuando con el ejemplo anterior tendríamos las 6 variables ordenadas

Ranking	Support
v3	91,50
v1	86,00
v2	86,00
v4	78,33
v5	76,50

En la primera iteración reducimos el conjunto de variables a 5

Supongamos q se obtiene una precisión de 85% para esta primera iteración.

Ranking	Support
v3	91,50
v1	86,00
v2	86,00
v4	78,33

En la segunda iteración reducimos a cuatro variables más y obtenemos una precisión de 90%.

Continuamos el algoritmo hasta obtener los resultados de cada subconjunto

Ranking	Support
v3	63
v3, v1	70
v3, v1, v2	93
v3, v1, v2, v4	81
v3, v1, v2, v4, v,5	90
v3, v1, v2, v4, v,5, v6	85

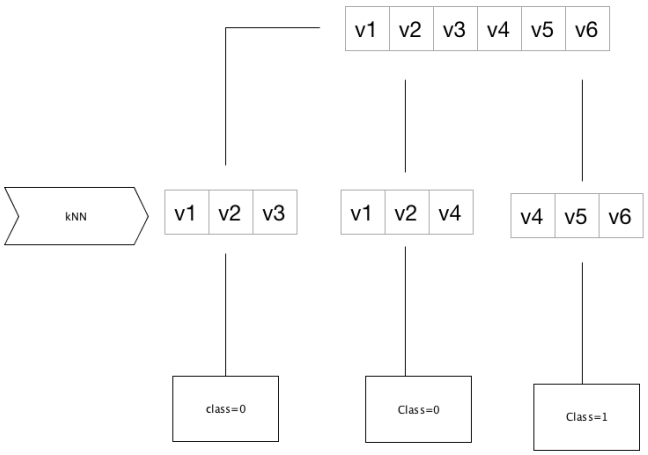
Así, podemos ver que el subconjunto de variables que mejor precisión presenta es {v3, v1, v2} con el 93% de precisión.

Resultados

Li, Harner y Adjerh obtuvieron una precisión del 91% sobre un conjunto de 34 muestras con una reducción de la dimensionalidad de 30 a 4 variables.

Construcción del modelo en base a la clasificación por RkNN.

Podemos construir un clasificador directo con el algoritmo RkNN aplicando una clasificación directa de los elementos del training set sobre una serie de conjuntos aleatorios de variables y resolviendo mediante votación la clase a asignar.



Así para un conjunto inicial de p variables podemos crear i conjuntos de j elementos (con i impar) aleatorios(con i y j pasados como parámetros) sobre los cuales aplicar kNN por separado como en el ejemplo del diagrama anterior.

A continuación se resolvería (análogamente a Random Forest) la pertenencia a determinada clase por votación.

La complejidad del diseño es mucho menor y el coste computacional se puede ajustar modificando los parámetros *i* y *j* para optimizar el compromiso entre rendimiento y precisión.

EJERCICIO 2

a) Ejecución del algoritmo.

```
runfile('/Users/luisarribas/Desktop/Aprendizaje computacional/PRA/Materiales/
knn_sklearn_pra.py', wdir='/Users/luisarribas/Desktop/Aprendizaje computacional/
PRA/Materiales')
                    * * * * * R E S U L T S * * * * *


[STANDARDIZATION PARAMETERS]
 - MEAN: [13.04 2.25 2.38 19.50 98.61 2.32 2.07 0.35 1.61 5.48 0.92 2.48 746.23]
 - STD: [0.84 1.00 0.28 4.25 12.54 0.69 1.11 0.13 0.54 2.41 0.24 0.74 295.01]


[CLASSES]
 - KNN            : [1 1 3 1 2 1 2 3 2 3 1 3 1 2 1 2 2 2 1 2 1 2 3 3 3 3 2 1 2 1
1 2 3 1 1 1 3
 3 2 3 1 2 2 3 3 1 2 2 3 1 2 1 1 3 3 2 1 1 2 1 3 2 1 3 1 1 1 3 1 1 1 3 2 1
 3 2 1 3 2 2 1 3 1 1 2 1 1 3 2 1 2 1 2 2 2 3 3 1 2 3 3 3 1 1 2 3 3 2 3 2 2
 2 1 1 3 1 3 1 1 2 3 1 1 1 1 1 2 3 2 2 2 3 3 2]
 - GROUND TRUTH   : [1 1 3 1 2 1 2 3 2 3 1 3 1 2 1 2 2 2 1 2 1 2 2 3 3 3 2 2 2 1
1 2 3 1 1 1 3
 3 2 3 1 2 2 2 3 1 2 2 3 1 2 1 1 3 3 2 2 1 2 1 3 2 2 3 1 1 1 3 1 1 2 3 2 1
 3 2 1 3 2 2 1 2 1 1 2 1 1 3 2 2 2 1 2 2 2 3 3 1 2 3 3 2 2 1 2 3 3 2 3 2 2
 2 1 1 3 1 3 1 1 2 2 1 1 1 2 1 2 3 2 2 2 3 3 2]


 - ACCURACY       : 91.04477611940298%


 * * * * * * * * * * * * * * * * * *
 - TOTAL ERRORS      : 12



 - CONFUSION MATRIX:      :

 [[45  0  0]
 [ 7 43  5]
 [ 0  0 34]]


 - EXECUTION TIME           : 0.005572795867919922


                        * * * * * E N D * * * * *
```

b) Comentarios sobre el código

La construcción del algoritmo se ha hecho reutilizando el código de *knn_sklearn.py.* Se ha añadido un bloque nuevo para el cálculo de los errores mediante la comparación item a item de Ground truth y la clasificación obtenida por knn, otro bloque para construir e imprimir la matriz de confusión y se ha implementado un control de tiempo para medir la ejecución del algoritmo.

El valor de *k* en la ejecución mostrada es 1.

c) Partición del conjunto de datos inicial en conjuntos de entrenamiento, pruebla y ground truth

Para la partición del conjunto de datos se ha utilizando un 75% como train set y un 25% como test set. Los datos deben barajarse (shuffle) para extraer el train set, ya que el fichero a clasificar está ordenado por clases, de manera que si se toma un bloque de items consecutivos del fichero con train set el resultado arroja resultados con un gran sesgo observacional.

Una vez separados los datos, el algoritmo crea los conjuntos *ground truth* con la columna class de *train set* y *test set* respectivamente. A continuación esa variable es eliminada de *test* y *train.*

***d*) Explicación del resultado**

Estandarización: el algoritmo realiza una estandarización del conjunto de datos para equilibrar el peso de las variables dada la dispersión de las magnitudes. Para estandarizar se resta la desviación típica y se divide por la varianza, de modo que el conjunto final tiene una media=0 y desviación estandar =1.

Classes (Knn y Ground truth): se listan respectivamente todos los items de *test dataset.* KNN indica la predicción que realiza el algoritmo y Ground truth la clase real, de modo que podamos medir los resultados.

Accuracy: en base al resultado anterior se calcula la precisión del algoritmo. A mayor tamaño del train set mayor la precisión.

Total errores: cantidad de clasificaciones erroneas hechas

Matriz de confusión: en esta matriz se situan en ambos ejes, ground truth y la predicción efectuada por knn, los posibles valores de la clasificación (1, 2 , 3) de modo que aquellos resultados distintos de cero que se encuentren fuera de la diagonal son errores en la predicción. La forma matricial permite ver la tendencia en los errores de clasificación. Así, el 58% de los errores son items erroneamente clasificados como clase 1 cuando son clase 2 (en rojo) y el resto de los errores (5) son clasificaciones como 3 cuando son de clase 2.

```
   1   2   3
1[45   0   0]
2[ 7 43   5]
3[ 0   0 34]]
```

EJERCICIO 3

Justificación

Partiendo de los algoritmos k Nearest Neibourghs y Random Forest (múltiples árboles de decisión) el algoritmos RkNN toma determinados conceptos de ambos modelos de clasificación para "hibridarlos" en un único algoritmo descrito extensamente en el paper "Random KNN feature selection - a fast and stable alternative to Random Forests. BMC Bioinformatics. 2011;12:450. doi: 10.1186/1471-2105-12-450" de Li, Harner y Adjerh.

La función del algoritmo implementado es la reducción de dimensionalidad en determinados problemas tipo "p grande, n pequeño" como hemos visto en el ejercicio 1.

Visión general del código propuesto

El código con la implementación se compone de dos bloques. Construcción del modelo y ejecución de prueba del modelo.

Cambios respecto al modelo propuesto en el artículo

El algoritmo implementado es el propuesto por los autores pero se han incluido algunas modificaciones:paso de parámetros de entrada {k, p, a , s, n} y eliminación de la primera fase, o fase geométrica de reducción de características.

Pasamos a describir los parámetros de entrada para la construcción del modelo:

k: número de vecinos más cercanos para el cálculo del algoritmo kNN

p: precisión máxima. Si se desea reducir el coste computacional en la ejecución del modelo puede limitarse la precisión máxima a la hora de su construcción., d emanera que la cantidad de atributos o variables quede reducida por este parámetro. Dicho de otro modo podemos pasar un parámetro p con 0<p<1 y el algoritmo encontrará y ordenará las n variables necesarias para conseguir esa precisión requerida descartando el resto.

a: número de variables. En lugar de limitar la precisión máxima podemos limitar el número de variables del modelo, de modo que este se construya con las a características más representativas.

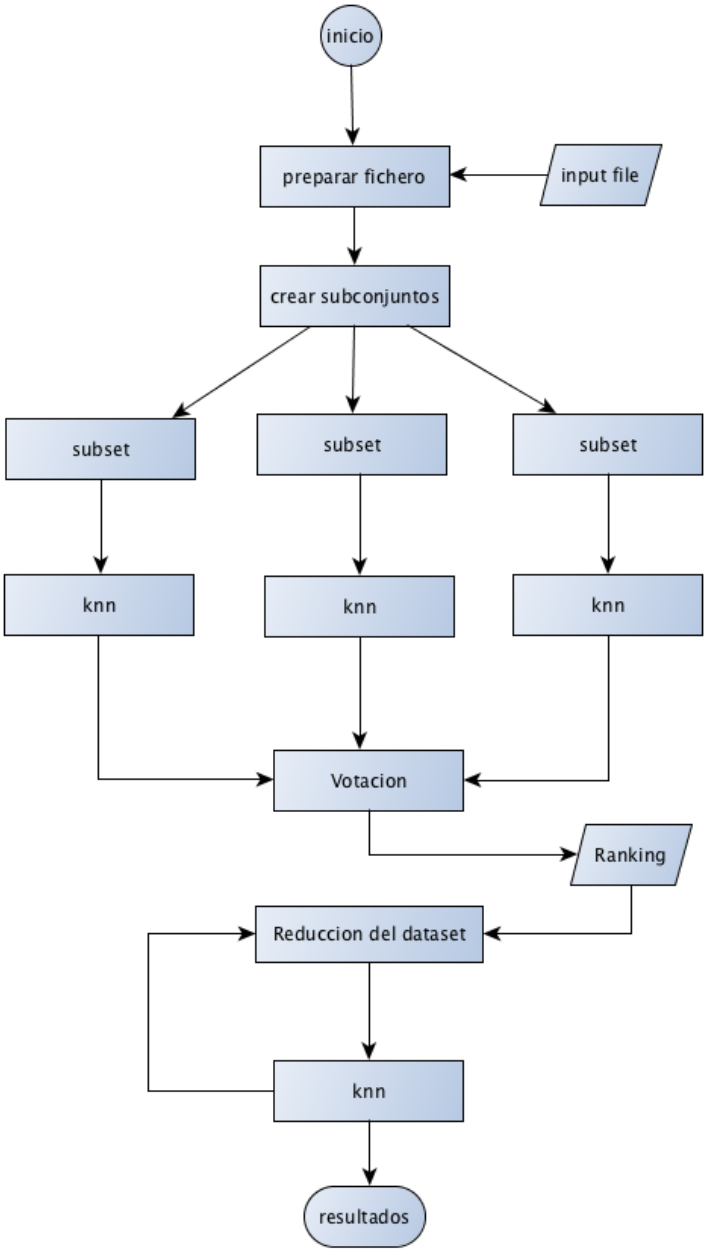
s: número de subconjuntos aleatorios para la construcción del modelo. A mayor número de subconjuntos, mayor fiabilidad obtendremos en el ranking de caractrísticas generado, ya que las votaciones para determinar la representatividad de cada atributo serán mayores. El coste computacional se incrementará a medida que se incrementes s, así como la fiabilidad.

n: atributos por cada subconjunto aleatorio creado. Podemos controlar la cantidad de atributos en cada sunbconjunto, de modo que podemos experimentar con diferents valores de n y s .

inputFile: fichero de entrada. A partir de este fichero de extensión .CSV se construye el modelo. Debe cumplir una serie de características comentadas ene el código.

Construcción del modelo.

A continuación vemos un diagrama de bloques del constructor.



EL algoritmo está descompuesto en un conjunto de funciones descritas cada una en los comentarios del propio código.

Consideraciones acerca de la fiabilidad de selección de características para el conjunto de datos propuesto

Debe tenerse en cuenta que el tamaño de p para el fichero proporcionado es notablemente pequeño y que además, según los resultados mostrados, la representatividad de las primeras

características clasificadas es bastante similiar, por lo que para repetidas ejecuciones con s=10.000 las 4 primeras posiciones del ranking varian, por lo que el modelo es más preciso para descartar las *m* variables menos representativas que para seleccionarn *n* características que describan exactamente el conjunto siendo n<<m

Explicación de los resultados presentados

Los resultados impresos en pantalla son, en primer lugar los de la constriucción del modelo y a continuación la ejecución del modelo con el fichero de entrada seleccionado.

El ranking -MOST REPRESENTATIVE FEATURES muestra ordenadas de mayor a menor representatividad los índices de columna del fichero .CSV pasado como parámetro. Así 1=ALCOHOL, 2=MALIC, 3= ASH, 4=ALCALINITY, 5=MAGNESIUM, 6= PHENOLS, 7=FLAVANOIDS, 8=NONFLAVANOIDS, 9=PROANTHOCYANIS, 10=COLOR, 11=HUE, 12=OD280_OD315, 13=PROLINE

Ejecución del algoritmo

Vemos a continuación los resultados de algunas ejecuciones con diferentes parámetros de entrada:

1ª prueba: {k, p, a , s, n}={1, 1, 13, 1000, 4}: con estos parámetros podríamos usar el algoritmo para clasificar de forma las 13 características por su relevancia.

```
runfile('/Users/luisarribas/Desktop/Aprendizaje computacional/PRA/Materiales/
knn_sklearn_pra_ex3.py', wdir='/Users/luisarribas/Desktop/Aprendizaje computacional/PRA/
Materiales')
          * * * * * R k N N   F E A T U R E   S E L E C T I O N   R E S U L T S * * * *

[MODEL PARAMETERS]

 - K= 1

 - MAXIUM ACCURACY:100%

 - RANDOM SUBSETS GENERATED: 1000

 - ATTRIBUTES PER SUBSET   : 4

 - MAXIMUM NUMBER OF ATTRIBUTES   : 13

[RESULTS]

 - MOST REPRESENTATIVE FEATURES      : [7, 1, 2, 3, 8, 4, 5, 10, 11, 12, 0, 9]
```

```
 - PREDICTED ACCURACY OF THE MODEL    : 93.28358208955224%


 - MODEL CONSTRUCTION TIME            : 0.925832986831665



          * * * * * R k N N    M O D E L    T E S T    R E S U L T S * * * *


[STANDARDIZATION PARAMETERS]
 - MEAN: [0.36 2.17 2.35 19.07 1.65 97.95 2.35 0.98 2.63 794.64 13.05 5.09]
 - STD: [0.12 1.21 0.29 3.17 0.53 10.65 0.57 0.21 0.69 336.20 0.79 2.45]


[CLASSES]
 - KNN            : [2 3 1 3 2 2 1 1 1 3 1 2 2 1 1 1 3 2 3 2 1 1 1 1 2 1 2 1 2 3 2 1 3 2 3
1 2
 3 3 2 1 1 2 3 2 3 2 3 3 3 2 1 3 2 1 2 3 1 1 1 3 3 2 1 2 3 3 2 2 2 3 1 2 2
 3 3 1 3 2 2 1 2 2 1 2 3 3 2 1 3 1 3 3 3 1 1 2 1 1 3 1 1 2 1 2 1 1 1 1 1 3
 1 1 2 3 1 3 2 2 2 2 3 2 2 1 1 3 3 3 2 3 3 1 3]
 - GROUND TRUTH   : [2 3 1 3 2 2 1 2 1 3 1 2 2 1 1 1 2 2 3 2 2 1 1 1 2 1 2 1 2 3 2 1 3 2 3
1 2
 3 3 2 1 2 2 3 2 3 2 3 3 3 2 2 3 2 1 2 2 1 1 1 3 3 2 1 2 3 3 2 2 2 3 1 2 2
 3 3 1 3 2 2 1 2 2 1 2 3 3 2 1 2 1 3 3 3 1 1 2 2 1 3 1 1 2 1 2 1 1 1 1 1 3
 1 1 2 3 1 3 2 2 2 2 3 2 2 1 1 3 3 3 2 3 2 1 2]


 - ACCURACY       : 92.53731343283582%


 - TOTAL ERRORS      : 10



 -CONFUSION MATRIX:      :

 [[43  0  0]
 [ 5 44  5]
 [ 0  0 37]]


 - MODEL EXECUTION TIME            : 0.004426002502441406


                      * * * * * E N D * * * *
```

2ª prueba: {k, p, a , s, n}={1, 0.75, 13, 1000, 4}: dada una precisión máxima del 75% buscamos el conjunto de atributos mínimo necesario.

Podemos ver que el modelo predice que en los 6 atributos listados tenemos el 70% de representatividad.

```
runfile('/Users/luisarribas/Desktop/Aprendizaje            computacional/PRA/Materiales/
knn_sklearn_pra_ex3.py',  wdir='/Users/luisarribas/Desktop/Aprendizaje  computacional/PRA/
Materiales')
          * * * * * R k N N    F E A T U R E    S E L E C T I O N    R E S U L T S * * * *
```

```
[MODEL PARAMETERS]

 - K= 1

 - MAXIUM ACCURACY:75.0%

 - RANDOM SUBSETS GENERATED: 1000

 - ATTRIBUTES PER SUBSET   : 4

 - MAXIMUM NUMBER OF ATTRIBUTES   : 13

[RESULTS]

 - MOST REPRESENTATIVE FEATURES      : [7, 2, 1, 8, 3, 4]

 - PREDICTED ACCURACY OF THE MODEL   : 70.8955223880597%

 - MODEL CONSTRUCTION TIME           : 0.9372930526733398


          * * * * * R k N N   M O D E L   T E S T   R E S U L T S * * * *

[STANDARDIZATION PARAMETERS]
 - MEAN: [0.36 2.40 2.36 1.61 18.87 101.93]
 - STD: [0.11 0.21 1.00 0.53 2.93 12.34]


[CLASSES]
 - KNN           : [3 1 2 2 3 1 2 3 3 3 1 3 1 1 3 2 2 3 1 1 1 3 2 2 2 3 1 2 2 3 1 2 3 1 2
1 3
 2 1 3 3 2 2 2 3 3 2 1 1 1 2 2 3 1 1 1 1 1 2 3 2 3 3 1 1 2 2 1 1 3 2 3 1 3
 2 2 2 2 1 2 2 1 1 3 3 1 2 3 3 2 3 2 3 3 2 3 3 1 2 2 3 1 2 2 3 1 1 1 1 3 2
 1 2 1 1 2 1 2 1 3 1 3 1 3 1 2 2 3 1 3 1 1 3 2]
 - GROUND TRUTH  : [2 1 2 2 3 2 2 3 2 3 1 3 3 1 2 2 2 3 1 1 1 3 1 2 2 3 1 2 3 3 1 2 3 1 3
1 3
 2 1 3 3 1 1 2 3 3 2 1 1 1 2 2 3 1 1 1 1 1 2 3 2 3 2 1 1 2 2 2 2 2 2 3 1 2
 2 2 2 3 1 2 3 1 1 2 3 1 2 3 2 2 2 1 3 3 2 2 3 1 2 1 3 1 2 2 3 1 2 1 2 3 2
 1 2 1 2 2 1 2 1 2 1 2 3 3 1 2 2 3 2 2 2 1 2 2]

 - ACCURACY      : 75.3731343283582%

 - TOTAL ERRORS     : 33


 -CONFUSION MATRIX:     :
```

```
[[37  5  0]
[ 8 36 14]
[ 2  4 28]]


- MODEL EXECUTION TIME            : 0.004198789596557617

                        * * * * * E N D * * * *
```

3ª prueba: (k, p, a , s, n)=(1, 1, 7, 1000, 4): pretendemos reducir los atributos a 7, la mitad del conjunto inicial propuesto.

Podemos ver en los resultados (en rojo) que el modelo predice que las 7 características del ranking[2, 7, 3, 8, 4, 5, 1] acumulan representatividad para dar una precisión del 71% .

Al ejecutarse el test sobre el modelo obtenemos un 73%.

```
runfile('/Users/luisarribas/Desktop/Aprendizaje computacional/PRA/Materiales/knn_sklearn_pra_ex3.py', wdir='/Users/luisarribas/Desktop/Aprendizaje computacional/PRA/Materiales')
      *****RkNN FEATURE SELECTION RESULTS****

[MODEL PARAMETERS]

 - K= 1

 - MAXIUM ACCURACY:100%

 - RANDOM SUBSETS GENERATED: 1000

 - ATTRIBUTES PER SUBSET   : 4

 - MAXIMUM NUMBER OF ATTRIBUTES   : 7

[RESULTS]

 - MOST REPRESENTATIVE FEATURES      : [2, 7, 3, 8, 4, 5, 1]

 - PREDICTED ACCURACY OF THE MODEL   : 71.64179104477611%

 - MODEL CONSTRUCTION TIME           : 0.96958327293396


      *****RkNN MODEL TEST RESULTS****
```

[STANDARDIZATION PARAMETERS]
- MEAN: [2.39 0.36 19.23 1.64 104.18 2.28 2.43]
- STD: [0.27 0.12 2.94 0.57 17.14 0.63 1.18]

[CLASSES]
- KNN : [1 1 2 2 2 3 2 1 3 1 2 1 1 2 3 2 3 1 3 1 1 3 1 2 1 1 1 3 2 1 3 2 2 1 1 1 2
2 1 3 3 1 3 1 1 1 3 3 1 3 3 1 2 2 2 2 2 2 1 3 2 3 1 1 1 2 3 1 1 1 3 1 1 3
2 1 1 3 2 3 2 1 3 2 1 3 3 1 3 3 2 3 3 2 2 1 3 1 3 2 1 3 1 1 3 1 1 1 2 2 1
1 1 1 3 2 2 2 1 2 3 1 1 3 1 2 1 1 2 1 3 2 1 1]
- GROUND TRUTH : [2 2 2 2 2 3 3 1 3 2 2 1 2 2 3 3 3 1 3 1 1 3 1 1 2 1 1 3 2 1 3 2 2 2 3 1 1 2
2 1 3 3 1 3 2 2 1 3 2 1 3 2 2 2 2 2 2 2 3 2 3 2 2 2 2 1 3 3 1 2 1 2 2 1 3
2 2 1 3 2 3 2 2 3 2 1 3 2 1 3 3 2 3 3 2 3 2 3 1 3 2 1 2 1 2 3 1 1 1 2 2 1
1 1 1 3 2 2 3 2 2 3 2 1 2 1 2 1 2 2 1 3 2 1 1]

- ACCURACY : 73.88059701492537%

- TOTAL ERRORS : 35

-CONFUSION MATRIX: :

[[38 1 0]
[20 31 7]
[1 6 30]]

- MODEL EXECUTION TIME : 0.00404810905456543

* * * * * E N D * * * *

.

Comparación con los resultados del ejercicio 2

RkNN feature selection tiene como función la clasificación de características. Eliminando las menos relevantes obtenemos mejoras en los tiempos de ejecución a costa de la precisión.

Como ejemplo tomamos el tiempo de ejecución de la prueba con las 7 caractrísticas más relevantes (prueba 3)

Vemos la comparación de ambos algoritmos en la siguiente tabla. La reducción de características de 13 a las 7 más representativas supone una mejora en el tiempo de ejecución del 27%, pero a costa de una pérdida de precisión del 17%

	kNN	**RkNN**	IMPROVEMENT
TIME	0,00557	0,004048	-27,32 %
ACCURACY	91,0447 %	73,8805 %	-17,1642 %