Inteligencia Artificial

Clasificadores con Weka
Parte avanzada

Roberto Matilla Augustinus

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

gti Grado en tecnologías interactivas

1. Parte 1: Predicción

1.1 Datos originales

```
 1 1:tested_negative
 2 2:tested_positive
 3 1:tested_negative
 4 2:tested_positive
 5 1:tested_negative
 6 1:tested_negative
 7 1:tested_negative
 8 1:tested_negative
 9 2:tested_positive
10 1:tested_negative
```

1.2 Modelo K-nn

```
=== Predictions on user test set ===

    inst#     actual  predicted error prediction
        1 1:tested_negative 1:tested_negative       0.999
        2 2:tested_positive 1:tested_negative   +   0.999
        3 1:tested_negative 1:tested_negative       0.999
        4 2:tested_positive 2:tested_positive       0.999
        5 1:tested_negative 1:tested_negative       0.999
        6 1:tested_negative 1:tested_negative       0.999
        7 1:tested_negative 2:tested_positive   +   0.999
        8 1:tested_negative 1:tested_negative       0.999
        9 2:tested_positive 2:tested_positive       0.999
       10 1:tested_negative 1:tested_negative       0.999

=== Summary ===

Correctly Classified Instances           8               80      %
Incorrectly Classified Instances         2               20      %
Kappa statistic                          0.5238
Mean absolute error                      0.2008
Root mean squared error                  0.4466
Total Number of Instances               10
```

Con la implementación de este modelo, podemos ver que se han cometido **dos fallos** respecto a los datos originales. Como podemos ver en la imagen superior, el **caso 2 y el caso 7** son resultados erróneos, por lo que se han clasificado un 80% de los casos de forma correcta y un 20% de forma incorrecta.

1.3 Modelo de Bayes

```
=== Predictions on user test set ===

    inst#     actual  predicted error prediction
        1 1:tested_negative 1:tested_negative       0.937
        2 2:tested_positive 2:tested_positive       0.971
        3 1:tested_negative 1:tested_negative       0.972
        4 2:tested_positive 2:tested_positive       0.934
        5 1:tested_negative 1:tested_negative       0.952
        6 1:tested_negative 2:tested_positive   +   0.683
        7 1:tested_negative 1:tested_negative       0.896
        8 1:tested_negative 1:tested_negative       0.927
        9 2:tested_positive 1:tested_negative   +   0.856
       10 1:tested_negative 1:tested_negative       0.975

=== Summary ===

Correctly Classified Instances           8               80      %
Incorrectly Classified Instances         2               20      %
Kappa statistic                          0.5238
Mean absolute error                      0.1975
Root mean squared error                  0.3505
Total Number of Instances               10
```

Con la implementación de este modelo, podemos ver que se han cometido **dos fallos** respecto a los datos originales, al igual que en el modelo anterior. Como podemos ver en la imagen superior, el **caso 6 y el caso 9** son resultados erróneos, por lo que se han clasificado un 80% de los casos de forma correcta y un 20% de forma incorrecta.

1.4 Modelo de árboles

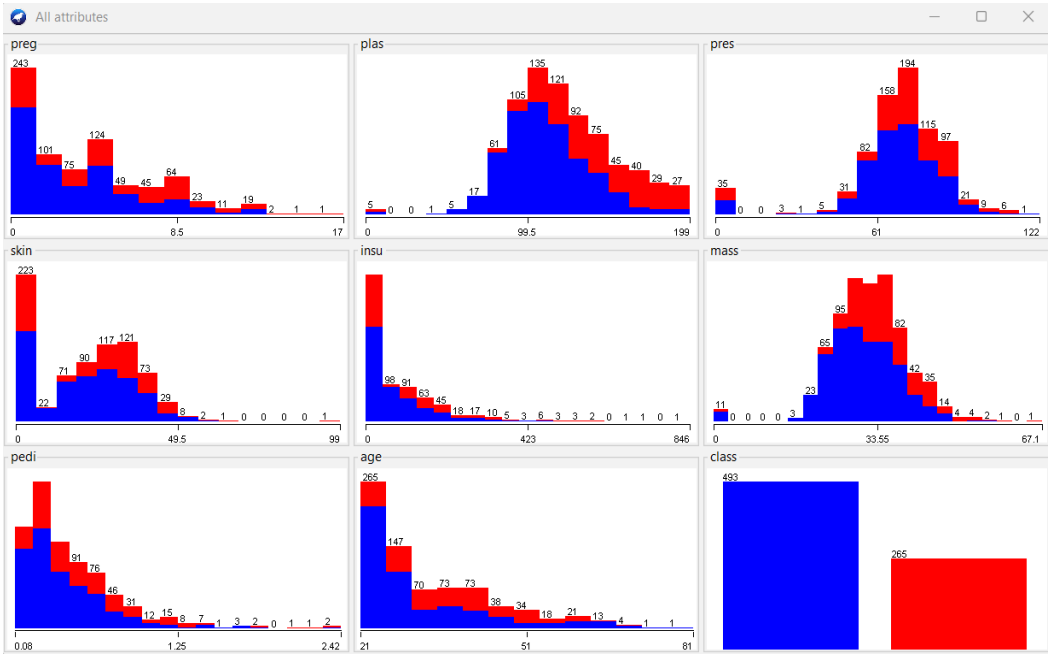
```
=== Predictions on user test set ===

    inst#     actual  predicted error prediction
        1 1:tested_negative 1:tested_negative       0.875
        2 2:tested_positive 2:tested_positive       0.867
        3 1:tested_negative 1:tested_negative       0.875
        4 2:tested_positive 2:tested_positive       0.867
        5 1:tested_negative 1:tested_negative       0.977
        6 1:tested_negative 1:tested_negative       0.598
        7 1:tested_negative 1:tested_negative       0.875
        8 1:tested_negative 1:tested_negative       0.977
        9 2:tested_positive 1:tested_negative   +   0.598
       10 1:tested_negative 1:tested_negative       0.875

=== Summary ===

Correctly Classified Instances           9               90       %
Incorrectly Classified Instances         1               10       %
Kappa statistic                          0.7368
Mean absolute error                      0.1813
Root mean squared error                  0.2486
Total Number of Instances               10
```

Con la implementación de este modelo, podemos ver que se ha cometido **solamente un fallo**, a diferencia de los dos anteriores modelos. Como podemos ver en la imagen superior, el erróneo ha sido el **caso 9**, haciendo así que se consiga un 90% de resultados correctos y un 10% de resultados erróneos.

Podemos deducir que este modelo es el más adecuado de los tres probados en esta práctica.

2 Parte 2: Parámetros

2.1 Visualización de todos los atributos. Archivo diabetes.arff



2.2 InfoGainAttributeEval y Ranker

```
=== Attribute selection 5 fold cross-validation (stratified), seed: 1 ===

average merit      average rank  attribute
 0.183 +- 0.006     1    +- 0       2 plas
 0.079 +- 0.007     2.8  +- 0.75    6 mass
 0.075 +- 0.009     3    +- 0.63    8 age
 0.063 +- 0.019     3.4  +- 1.2     5 insu
 0.045 +- 0.012     4.8  +- 0.4     1 preg
 0.015 +- 0.013     6.8  +- 0.98    7 pedi
 0.008 +- 0.01      7    +- 0.63    3 pres
 0.016 +- 0.015     7.2  +- 0.75    4 skin
```

Como sabemos, el método **Ranker** ordena los atributos según su importancia, y nos muestra dos resultados diferenciados: **Average Merit** y **Average Rank**, como podemos ver en la imagen superior.

Comparando los datos de la imagen con la imagen del Apartado 1, fijándonos en **Average Rank**, podemos relacionar los resultados de la siguiente manera:

- El atributo **plas** tiene una desviación típica de 0 y está situado en la posición 1. Esto significa que en los 5 ciclos realizados, su posición no ha variado nada, considerando entonces que es el atributo con más importancia. Comparando con su gráfica, podemos corroborar estos datos ya que es la que mejor diferenciadas tiene las clases.

- Después del atributo plas, están los atributos **mass**, **age** e **insu**, que le siguen en cuanto a posición. En cuanto a la desviación típica de estos atributos, podemos ver que ronda cercana al uno, y que su nivel de importancia sigue siendo bastante alto. Esto tiene sentido comparándolo con las gráficas, ya que después del atributo **plas**, desde mi punto de vista hay gran diferenciación en cuanto a las clases en estas.

- Finalmente tenemos los atributos **preg**, **pedi**, **pres** y **skin**, que no consiguen un buen resultado, al igual que su diferenciación de clases en las gráficas.

2.3 CfsSubsetEval y GreedyStepwise

```
=== Attribute selection 5 fold cross-validation (stratified), seed: 1 ===

number of folds (%)  attribute
           0(  0 %)    1 preg
           5(100 %)    2 plas
           0(  0 %)    3 pres
           0(  0 %)    4 skin
           1( 20 %)    5 insu
           5(100 %)    6 mass
           3( 60 %)    7 pedi
           5(100 %)    8 age
```

En este caso, sabemos que **CfsSubsetEval** evalúa el valor de un subconjunto de atributos considerando la capacidad predictiva individual de cada característica junto con el grado de redundancia entre ellas, y nos muestra el porcentaje de veces que fueron seleccionados en los ciclos seleccionados.

Como podemos ver, los atributos **plas**, **mass** y **age** han sido seleccionados en todos los ciclos realizados, dando a entender que estos atributos son los mejores en cuanto a su criterio de funcionamiento. Estos resultados reafirman lo que se mostró en el método anterior, siendo también estos tres atributos los mejores.

Después de estos atributos, vemos que les siguen el **pedi** (60%) e **insu** (20%), como los mejores. Podemos observar que su porcentaje de selección es bastante inferior, y de nuevo, vuelve a reafirmar lo visto anteriormente y comparándolo con las gráficas obtenidas.

Finalmente, los atributos **preg**, **pres** y **skin** no han sido seleccionados en ninguno de los ciclos, queriendo decir que estos atributos no tienen la suficiente importancia según el criterio.