Inteligencia artificial: Práctica 8

Álvaro García Tenorio [*] Miguel Pascual Domínguez[**]

13 de mayo de 2018

Índice general

[*] alvgar14@ucm.es
[**] miguepas@ucm.es

Capítulo 1

Agrupamiento

1.1. Descripción del conjunto

Para este experimento hemos seleccionado un conjunto de datos acerca de las características de 3 tipos de semillas diferentes. Los datos pertenecen a un estudio académico realizado por diversos individuos pertenecientes a las siguientes instituciones; la Universidad de Ciencias de la Computación y Matemáticas de Polonia, la Academia de Ciencias de Polonia y la Universidad de Tecnología de Polonia.

Los datos los hemos obtenido de la siguiente pagina, http://archive.ics.uci.edu/ml/datasets/seeds. El archivo descargado no venía en formato .arff, por tanto, hemos tenido que añadir parte del codigo de definición de atributos y de la relación, ya que solamente venía con los datos del estudio. Los datos no se han visto alterados por nuestras modificaciones. Expliquemos ahora los atributos/variables de este conjunto de datos:

1. Area: área de la semilla, número real.
2. Perimeter: perímetro de la semilla, número real.
3. Compact: compacidad de la semilla se calcula con la siguiente fórmula $C = 4 * \pi * Area/perimeter^2$, número real.
4. LengthKernel: longitud del centro de la semilla, número real.
5. Width: anchura del centro de la semilla, número real.
6. AssymmetryCoefficient: coeficiente de asimetría, número real.
7. LengthKernelGrove: longitud del la curvatura del centro de la semilla, número real.
8. VarietiesOfWheat: variedades de semilla de trigo: 1-Kama, 2-Rosa and 3-Canadian, variable categórica.

1.2. Parametrización del algoritmo de agrupamiento

Veamos ahora cómo preparamos los datos y las variables para aplicar el algoritmo de agrupamiento jerárquico.

En primer lugar, consideremos si debemos normalizar o estandarizar las variables. Para ello, basta con ejecutar el algoritmo jerárquico de agrupamiento y observar que manipula distancias, por tanto, lo recomendable es normalizar todas las variables numéricas para que así todas ellas trabajen en el mismo rango.

Decidamos ahora el número de clusters que queremos para nuestro agrupamiento, para ello, tenemos que observar qué ocurre dependiendo de qué variable tomemos como principal. Dado que todas las variables menos una son numéricas (siendo la otra categórica) se debe tomar la categórica como principal al realizar el clustering con la opción de "*Classes to clusters evaluation*", con lo cual el número de clusters óptimo es el mismo que el número de clases tiene la variable categórica, que en este caso es 3.

Para elegir el tipo de enlace que se usa en el algoritmo, hemos probado el algoritmo jerárquico con los 3 posibles enlaces, simple, centrado y completo; y el enlace con el que los resultados del algoritmo salían homogéneos y dignos de estudiar, era con el enlace centroide lo cual se debe a que los individuos del conjunto de datos pertenecientes a la misma clase están muy concentrados.

1.3. Resultados y conclusiones

Mostremos primero el sumario de weka en la figura 1.1, tras esto, mostramos los histogramas de weka en los que hemos añadido una variable llamada "cluster", que nos permite observar cómo estan organizados los clusters a lo largo de todas las variables, (idea obtenida de las recomendaciones y tutoriales de la página de weka); en la Figura 1.2.

Comentemos ahora algunos detalles del sumario, se puede observar que el reparto de las instancias entre los clusters, es realmente equitativo, pues dos clusters tienen un 32 % de las instancias y el otro tiene el 36 % de las instancias, que hacen un total del 100 %. Además solo se produce un error del 8 % en lo que consiste en la asignación de las instancias al correspondiente clusters, ya que observándolo en el sumario vemos que cada cluster corresponde con una tipo de semilla de trigo. El cluster 0 corresponde con la semilla de tipo 1, el cluster 1 con la semilla de tipo 2 y el cluster 2 con la semilla de tipo 3.

Veamos ahora variable por variable, qué valores toman en cada cluster, esto lo podemos obtener con la ayuda de los histogramas, pues cada cluster, 0,1 y 2, es un color, azul oscuro, rojo y azul claro.

Comencemos con el área, en este caso tenemos que los clusters se distribuyen de una manera gradual, ya que las instancias pertenecientes al cluster 2 están situadas en los valores mas inferiores con unas pocas instancias del cluster 0, después en los valores medios están las instancias del cluster 0 con algunas pocas del cluster 1 y los valores mas altos son todos de instancias pertenecientes al cluster 1.

El perímetro tiene una distribución equivalente, los del cluster 2, valores bajos; el cluster 0, unos pocos valores bajos y la mayoría valores medios; y el cluster 1, sobre todo valores grandes y unos pocos con valores medios.

La compacidad de la semilla está mas distribuida pues el cluster 2 tiene los valores menores, pero el resto se reparte de manera equitativa entre el cluster 0 y el 1, no obstante, predominan los del cluster 1 en los valores altos.

La longitud del centro tiene la siguiente distribución. El cluster 2 está junto con el cluster 0 en los valores mas bajo, y el cluster 1 ocupa los valores mas altos.

La anchura del centro tiene una distribución similar a la del perímetro y el área.

Con el coeficiente de simetría se produce el efecto inverso, como si cogieras el histograma y lo pusieras en un espejo, es decir, los clusters 0 y 1 ocupan los valores mas bajos, y el cluster 2 ocupa los valores mas altos, a excepción de dos valores del cluster 0, que están en los valores más altos, de lo cual se podría deducir que son las dos instancias de la clase 3 que están colocadas mal en los clusters, y en vez de estar en el cluster 2, están en el 0.

```
Clustered Instances

0       67 ( 32%)
1       76 ( 36%)
2       67 ( 32%)


Class attribute: VarietiesOfWheat
Classes to Clusters:

  0  1  2  <-- assigned to cluster
 60  8  2 | 1
  2 68  0 | 2
  5  0 65 | 3

Cluster 0 <-- 1
Cluster 1 <-- 2
Cluster 2 <-- 3

Incorrectly clustered instances :       17.0      8.0952 %
```

Figura 1.1: Sumario Clustering/Agrupamiento

Por ultimo, la curvatura del centro tiene la misma distribución que la longitud del centro.

Ahora queda por comentar la variable que evaluábamos, la variedad del trigo, y asi poder observar que cluster es mas homogeneo y cual tiene mas valores.

Con una cantidad de individuos de 76, tenemos que el cluster 1 es el cluster más numeroso a pesar de ser por pocos individuos pues no son mas de 10 individuos con ninguno de los otros dos clusters.

Además podemos observar que el cluster más homogéneo es el cluster 1 pues sus individuos solo pertenecen a dos clases la 1 y la 2, y además posee casi todos los elementos de la clase 2. Por su parte, el cluster mas inhomogeneo es el cluster 1 pues esta repartido entre las 3 clases.

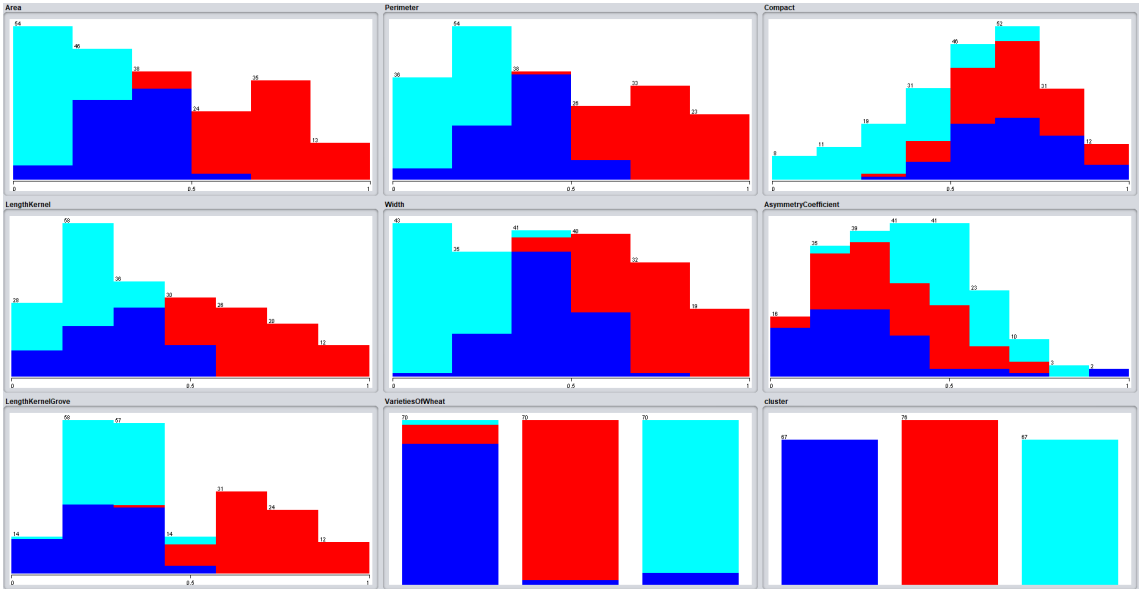

Figura 1.2: Histogramas Clustering

Capítulo 2

Clasificación

2.1. Descripción del conjunto

Para este tipo de problema, lo interesante es coger un conjunto o "dataset" que tenga pocos atributos y muchas instancias, por ello, el conjunto que hemos seleccionado es un dataset sobre una evaluación de coches, hemos obtenido los datos de la siguiente página http://archive.ics.uci.edu/ml/datasets/Car+Evaluation. Al igual que en el primer apartado, los archivos descargados no venían en formato .arff, por tanto hemos tenido que añadir parte del codigo de definición de atributos y de la relación, pero los datos no han sido modificados.

Expliquemos ahora los atributos y variables, todos ellos categóricos, ninguno numérico; de este conjunto de datos:

1. Compra: Precio de compra; muy alto (vhigh), alto (high), medio (med) y bajo (low).
2. Mantenimiento: Precio de mantenimiento; muy alto (vhigh), alto (high), medio (med) y bajo (low).
3. Puertas: Número de puertas clasificado en 4 categorías; 2, 3, 4 y 5 o más.
4. Personas: Número de personas clasificado en 3 categorías; 2, 4 y más.
5. Maletero: Capacidad del maletero; pequeña, media y grande.
6. Seguridad: baja, media y alta.
7. Tipo de coche según aceptabilidad: inaceptable (unacc), aceptable (acc), bueno (good) y muy bueno (vgood).

Una vez comentados los atributos, expliquemos el problema.

Tal y como son los datos, hemos decidido que para aprovechar el potencial de los mismos, lo más lógico es plantear el problema de clasificar la aceptabilidad de los coches en función de los otros 6 atributos, para así poder observar cuales coches son inaceptables y asi no venderlos o comprarlos; ya que así podemos observar más adelante el poder del algunos atributos como la seguridad y el número de personas que ya clasifican una cantidad considerable de los coches.

Mostramos aquí el histograma de la variable tipo de coche según aceptabilidad en la Figura 2.1.

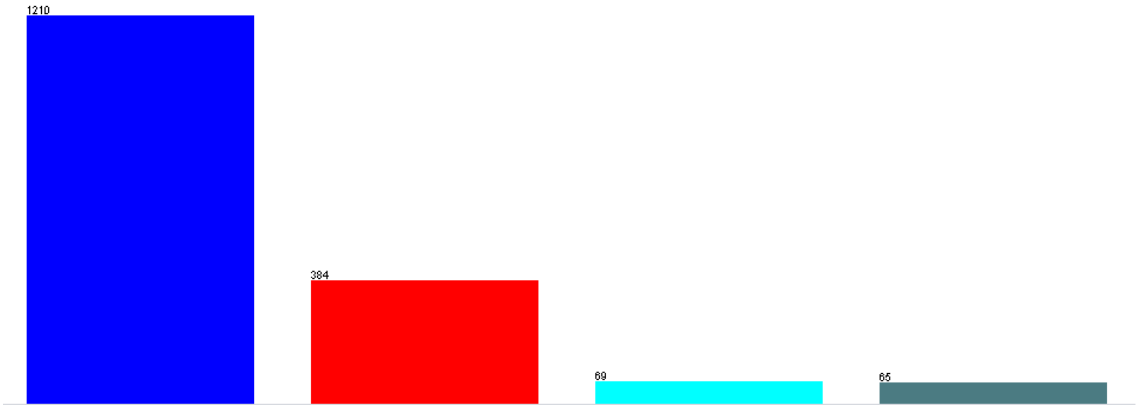

Figura 2.1: Histograma del tipo de coche según aceptabilidad

2.2. Parametrización del J48

Dado que todas nuestras variables son categóricas, no se las puede normalizar (englobar los valores entre 0 y 1) ni estandarizar (modificar los datos para que tengan media 0 y varianza 1). Primero se realizará una ejecución del algoritmo sin validación ni entrenamiento, después se realizará otra ejecución en validación y entrenamiento al 66 %.

En la primera ejecución se han mantenido los parámetros del algoritmo predeterminado por Weka y en la segunda ejecución al realizar un entrenamiento y una validación al 66 %, vimos que el árbol resultante era muy parecido al original sin entrenamiento ni validación, por tanto decidimos añadir una nueva peculiaridad a esta segunda ejecución y era que de manera forzada siempre se realizaran particiones binarias en el árbol, es decir que formara un árbol binario.

2.3. Resultados y conclusiones

Mostremos primero los resultados de la primera ejecución, primero mostramos el sumario de weka en la Figura 2.2, en el que esta incluido los valores del recall, precisión los errores y la matriz de confusión; y después el árbol en la Figura 2.3.

Mostraremos ahora los resultados de la segunda ejecución, primero mostramos el sumario de weka en la Figura 2.4, en el que esta incluido los valores del recall, precisión los errores y la matriz de confusión; y después el árbol en la Figura 2.5.

Comentemos ahora los resultados obtenidos en ambas ejecuciones:

Como se puede observar, en la primera ejecución se produce un árbol más o menos claro, en el que casi todos los nodos tienen 3 o 4 hijos, también se observa que el algoritmo le cuesta diferenciar entre los tipos de coches que son aceptables, buenos o muy buenos; tal y como se ve en la matriz de confusión, también se observa que el porcentaje de instancias mal clasificadas es realmente bajo pues es menor de un 5 %, lo cual es un porcentaje realmente bueno.

En la segunda ejecución, decidimos forzar a al árbol a que fuera binario, lo que provoca que se cree un árbol con mayor profundidad y menos homogéneo que el anterior, pero, en cambio, al realizar una parte de entrenamiento y validación, el porcentaje de fallos en la clasificación es menor, en unas pocas décimas que en la primera ejecución, y además, se observa en la matriz de confusión que la confusión (valga la redundancia) que había con los coches buenos y muy buenos ha desaparecido.

```
=== Summary ===

Correctly Classified Instances        1664               96.2963 %
Incorrectly Classified Instances        64                3.7037 %
Kappa statistic                          0.9198
Mean absolute error                      0.0248
Root mean squared error                  0.1114
Relative absolute error                 10.8411 %
Root relative squared error             32.9501 %
Total Number of Instances             1728

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
               0,977     0,019     0,992       0,977   0,984      0,948    0,997     0,998     unacc
               0,964     0,028     0,907       0,964   0,934      0,916    0,996     0,978     acc
               0,826     0,007     0,838       0,826   0,832      0,825    0,997     0,916     good
               0,846     0,003     0,917       0,846   0,880      0,876    0,999     0,952     vgood
Weighted Avg.  0,963     0,020     0,964       0,963   0,963      0,933    0,997     0,989

=== Confusion Matrix ===

    a    b    c    d   <-- classified as
 1182   25    3    0 |    a = unacc
   10  370    2    2 |    b = acc
    0    9   57    3 |    c = good
    0    4    6   55 |    d = vgood
```

Figura 2.2: Sumario Primera Ejecución

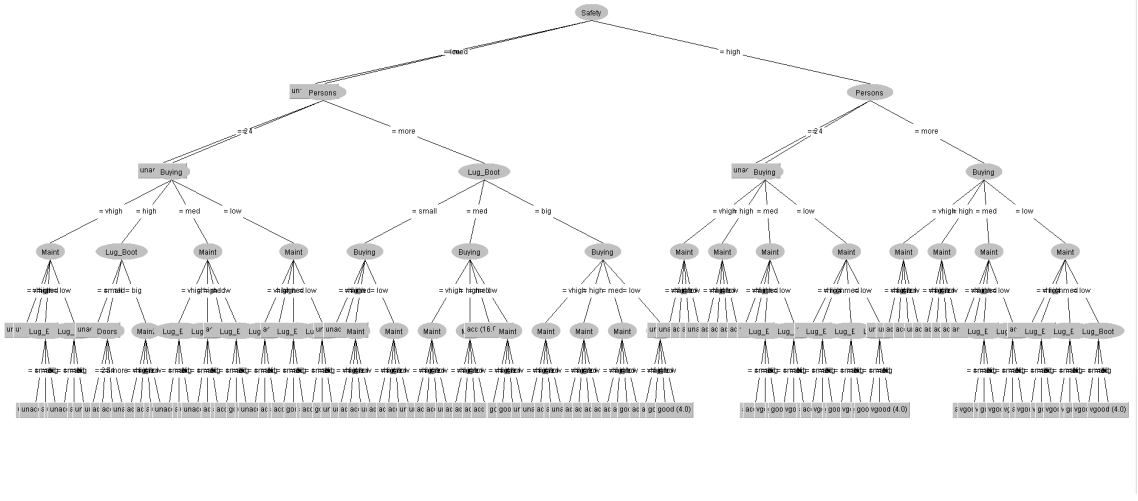

Figura 2.3: Árbol Primera Ejecución

```
=== Summary ===

Correctly Classified Instances         569               96.7687 %
Incorrectly Classified Instances        19                3.2313 %
Kappa statistic                          0.9304
Mean absolute error                      0.0227
Root mean squared error                  0.1204
Relative absolute error                  9.9006 %
Root relative squared error             35.5231 %
Total Number of Instances              588

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,980    0,022    0,990      0,980    0,985      0,952    0,993     0,997     unacc
                 0,942    0,013    0,956      0,942    0,949      0,933    0,990     0,963     acc
                 0,870    0,009    0,800      0,870    0,833      0,827    0,996     0,850     good
                 1,000    0,007    0,826      1,000    0,905      0,906    0,998     0,881     vgood
Weighted Avg.    0,968    0,019    0,969      0,968    0,968      0,941    0,993     0,980

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
 401   6   2   0 |   a = unacc
   4 129   3   1 |   b = acc
   0   0  20   3 |   c = good
   0   0   0  19 |   d = vgood
```

Figura 2.4: Sumario Primera Ejecución

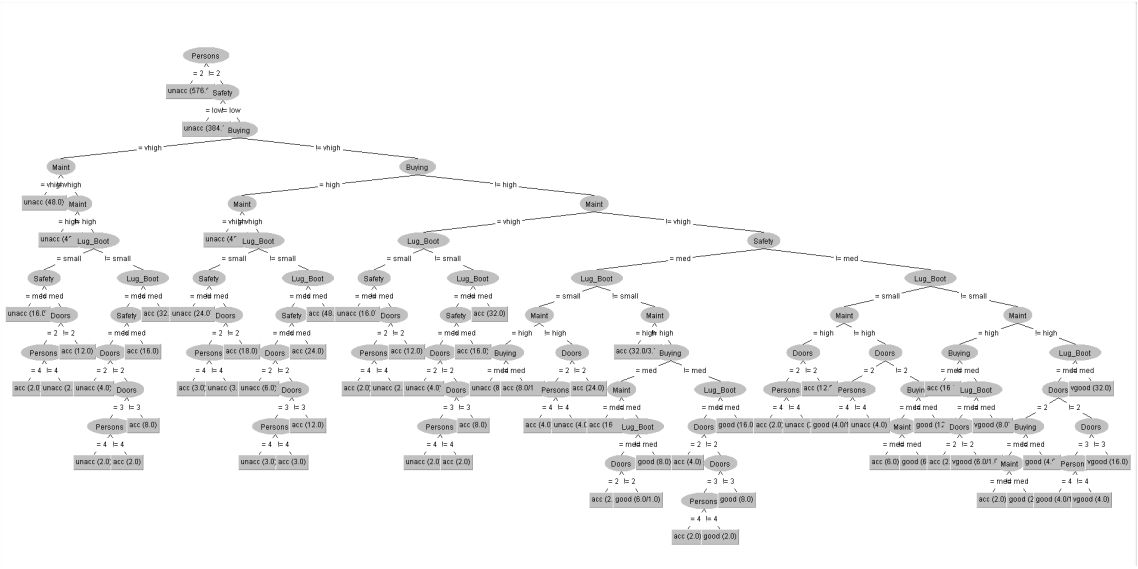

Figura 2.5: Árbol Primera Ejecución

En cambio, cabe destacar que, a pesar de que la segunda ejecución presenta menos confusión, la precisión que posee en los coches buenos, es menor que en la primera ejecución, a pesar de que la segunda tiene una precisión global mayor que en la primera ejecución. Además, la segunda parametrización posee una mayor sensibilidad a la hora de concretar en los tipos de coches según aceptabilidad y como es un valor alto y cercano a 1, sabemos que la validación y ambas ejecuciones son fiables.

Ahora respondamos a las preguntas planteadas.

Por la claridad, y la manera de filtrar que realiza, consideramos que es mejor árbol el de la primera ejecución, a pesar de que se confunde en algunas instancias, pero, para interpretar el problema, consideramos que se obtiene una interpretación mejor que con la segunda ejecución. Las principales diferencias se encuentran en las ramas iniciales, al ser el primero un árbol con una raíz con mas de dos hijos, hay más claridad y orden en el árbol.

En cambio, en términos de validación consideramos que el segundo árbol es más exacto y con mejores estadísticas que la primera ejecución.

Interpretemos ahora el primer árbol para responder las preguntas en función del contexto del problema.

La raíz del árbol evalúa el atributo de seguridad de un coche, la cual es una de las preguntas más importantes a la hora de valorar si un coche es aceptable o no, pues un coche inseguro es, desde nuestro punto de vista, uno de los vehículos más peligrosos. Por ello, como se puede observar, los coches que mejor ha clasificado y acertado han sido los coches inaceptables (unacc).

Como hemos comentado, la seguridad ha sido una de las variables mas relevantes en el problema, además de que es el nodo que más poder discriminante tiene, y, a pesar de esto, en la segunda ejecución no se ha tomado la seguridad en la raíz sino que se ha tomado el número de personas, a pesar de que están en el mismo nivel, esto se puede deber al algoritmo de selección interno de weka, y que, al exigirle la distinción binaria, hace que el número de personas tenga un mayor poder discriminante.

En efecto, se puede observar que hay dos variables que se confunden sobretodo entre sí, que son las dos relacionadas con el precio, buying y maint; a lo largo de todo el árbol pues normalmente, son las clases que menos discriminan y se seleccionan las últimas,y que además están organizados con los mismos valores, muy alto (vhigh), alto (high), medio (med) y bajo (low). Tiene cierto sentido que se confundan, pues normalmente un coche que cueste mucho dinero, también va a tener un coste de mantenimiento alto.

Además, desde nuestro punto de vista, estas dos clases, junto con la del tamaño del maletero, son las clases que menos información aportan, pues, al final, son las que clasifican entre los buenos coches, que si enfocamos este problema simplemente para descartar los coches peligrosos/inaceptables, dichas clases no nos darían información relevante.

Capítulo 3

Regresión

3.1. Descripción del conjunto

En el caso de la regresión, hemos decidido coger un conjunto de datos cuyas variables sean todas numéricas para así facilitar la ejecución y parametrización del algoritmo del perceptrón multicapa.

Por ello hemos seleccionado un conjunto de datos formado por los caracteristicas que determinan la resistencia del cemento, ya que todos los datos son numéricos. Los datos los hemos obtenido de la siguiente página http://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength.

En este tercer caso, ocurrió lo mismo que en los anteriores y tuvimos que añadir la parte de los atributos y ponerlo en formato de archivo arff.

Expliquemos ahora los atributos del problema, que en este caso, como hemos dicho antes, son todos numéricos:

1. CementQuantity: cantidad de cemento, número real.
2. BlastFurnaceSlagQuant: cantidad escoria de alto horno, conglomerado que se desprende de la fusión de hierro en un alto horno, número real.
3. FlyAshQuant: cantidad de ceniza voladora, producto que surge de la combustión del carbón, número real.
4. WaterAmount: cantidad de agua, número real.
5. SuperplasticizerAmount: cantidad de superplastificante, aditivo que reduce el contedido en agua del determinado hormigón, número real.
6. CoarseAggregateQuant: cantidad de grava fina, número real.
7. FineAggregateQuant: cantidad de gravilla, número real.
8. Age: edad del cemento, número real.
9. ConcreteCompressiveStrength: resistencia de la compresión del cemento, número real.

El propio enlace nos proporciona ya cuál debe ser nuestra variable de salida en el problema que hemos comentado y se trata de la última, la ,aquí presentamos el histograma con su distribución en la Figura 3.1.

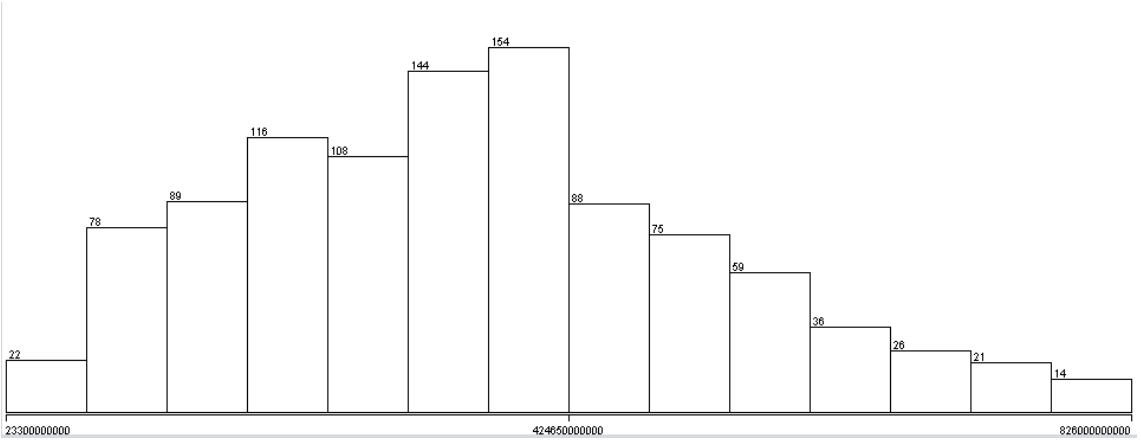

Figura 3.1: Histograma de la .

3.2. Parametrización del perceptrón multicapa y del K-NN

Primero debemos analizar si las variables requieren de normalización, estandarización, o, simplemente, dejarlas como están. Al probar con ambas opciones, normalizando y sin normalizar, sale con el algoritmo perceptrón multicapa los mismos resultados, por tanto, hemos optado por no normalizar, para así tener posibilidad de hacerlo con algún subconjunto de variables en la parametrización de K-NN, en cambio, si hemos optado por estandarizar los valores, para así disminuir un poco los errores que se comenten.

Después, asignemos el número de partes, p, en las que se realizará la validación cruzada para el perceptrón multicapa y para el K-NN, sabiendo que puede producirse un cambio con respecto al p en el K-NN. Hemos decidido que el más adecuado es $p = 20$, ya que, a base de pruebas, hemos encontrado que el error es menor con 20 que con algunos números menores y mayores a él.

El perceptrón multicapa no tiene ninguna parametrización más, en cambio, en el caso del K-NN, hay que determinar el valor de k, que en este caso para encontrar un error inferior o parecido al error que te proporciona el perceptrón multicapa, se ha tenido que tomar el valor $k = 1$, pues además si se aumentaba el k, el error aumentaba, lo cual eso no nos beneficiaba.

3.3. Resultados y conclusiones

Mostremos ahora el sumario del algoritmo del perceptrón multicapa en la Figura 3.2, en él se puede observar los errores.

Analicemos los errores, en el sumario, es fácil ver, que con este conjunto el error tanto la media del error absoluto como el de la raíz cuadrada de la raíz, son errores altos para este conjunto, al igual que se ve también en los errores relativos que tiene un porcentaje de 48 % y de 50 %.

Mostremos ahora el sumario del algoritmo K-NN en la Figura 3.3, en él, se pueden observar unos errores realmente parecidos a los del anterior sumario. En este caso se observa que en la media del error absoluto se obtiene un valor algo menor pero también sigue siendo un error alto, en cambio en la raíz cuadrada de la raíz se obtiene un error mayor, lo que indica que los valores más fuertes pesan que los mas pequeños y hace que el error suba con respecto al perceptrón multicapa.

```
=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.871
Mean absolute error               65195487574.7819
Root mean squared error           83800966150.6931
Relative absolute error                 48.3808 %
Root relative squared error             50.1345 %
Total Number of Instances            1030
```

Figura 3.2: Sumario del algoritmo del perceptrón multicapa.

```
=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.8648
Mean absolute error               62502686084.1424
Root mean squared error           87406560859.7985
Relative absolute error                 46.3825 %
Root relative squared error             52.2915 %
Total Number of Instances            1030
```

Figura 3.3: Sumario del algoritmo K-NN.