

0. Objetivo

Encontrar un buen modelo de clasificación que prediga la clase de los datos

1. Análisis de datos

1.1 Atributos

El dataset se compone de 1500 filas con 20 atributos. 19 de los atributos son numéricos y el último que es la clase de la clasificación es de texto:

- % 1. region-centroid-col: the column of the center pixel of the region.
- % 2. region-centroid-row: the row of the center pixel of the region.
- % 3. region-pixel-count: the number of pixels in a region = 9.
- % 4. short-line-density-5: the results of a line extractoin algorithm that counts how many lines of length 5 (any orientation) with low contrast, less than or equal to 5, go through the region.
- % 5. short-line-density-2: same as short-line-density-5 but counts lines of high contrast, greater than 5.
- % 6. vegde-mean: measure the contrast of horizontally adjacent pixels in the region. There are 6, the mean and standard deviation are given. This attribute is used as a vertical edge detector.
- % 7. vegde-sd: (see 6)
- % 8. hedge-mean: measures the contrast of vertically adjacent pixels. Used for horizontal line detection.
- % 9. hedge-sd: (see 8).
- % 10. intensity-mean: the average over the region of $(R + G + B)/3$
- % 11. rawred-mean: the average over the region of the R value.
- % 12. rawblue-mean: the average over the region of the B value.
- % 13. rawgreen-mean: the average over the region of the G value.
- % 14. exred-mean: measure the excess red: $(2R - (G + B))$
- % 15. exblue-mean: measure the excess blue: $(2B - (G + R))$
- % 16. exgreen-mean: measure the excess green: $(2G - (R + B))$
- % 17. value-mean: 3-d nonlinear transformation of RGB. (Algorithm can be found in Foley and VanDam, Fundamentals of Interactive Computer Graphics)

% 18. saturatoin-mean: (see 17)

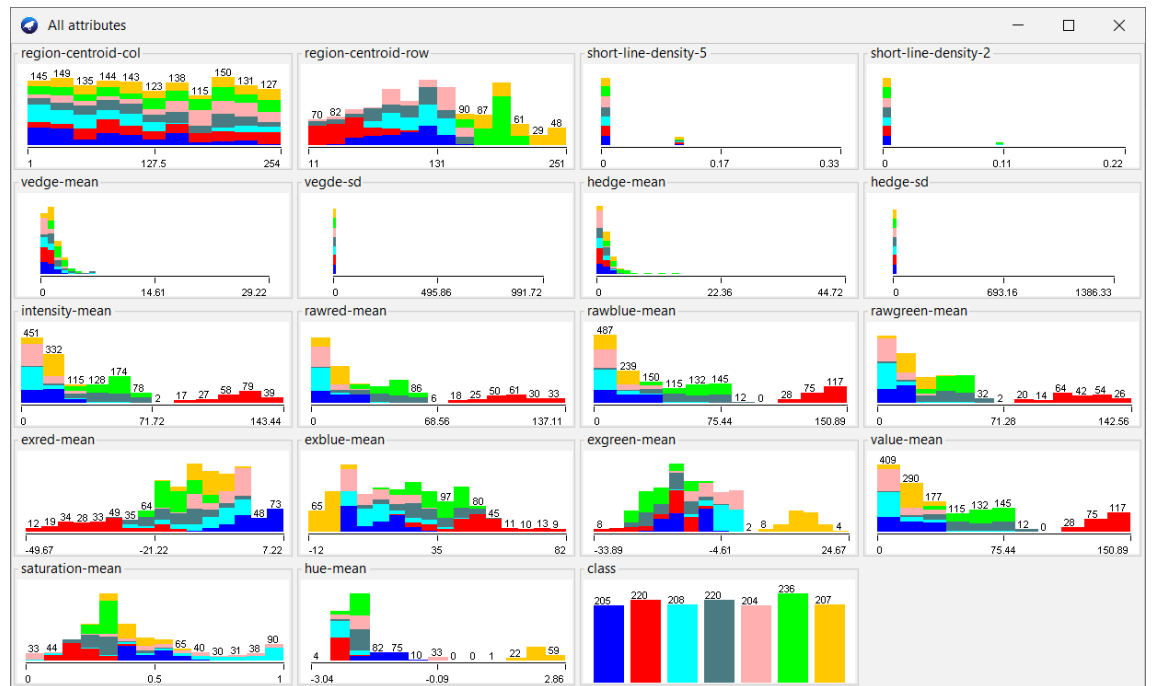
% 19. hue-mean: (see 17)

1.2 Análisis de los atributos: búsqueda de nulos, atributos que no aporten información al modelo, etc...

La columna region-pixel-count tiene el mismo valor para todas las files, por lo que procedo a eliminarla

No existent valors NA en el data set

El histograma del resto de columnes es este:



1.3 Se puede observar que las clases de clasificación están bastante compensadas. Hay aproximadamente el mismo número de muestras de cada classe

1.4 Algunos atributos como vegde-sd y hedge-sd parece que tienen un solo valor, aunque visto el histograma en detalle existent unos pocos valores diferentes

2. División de los datos para traint y test

2.1 Utilizaré una división de 80% para traint y 20% para test es decir 1200 instancias de traint y 300 de test

3. Prueba de diferentes modelos:

Para valorar los modelos voy a fijarme en primera instancia en el valor que da Correctly Classified Instances sobre las 300 instancias de test. Cuanto mayor sea el porcentaje mayor será el modelo. Si los valores son parecidos pasaré a examinar la matriz de confusión

3.1 Random Tree

3.1.1 Con todos los valores por defecto:

De las 300 instancias clasifica bien 287 => 95.67 % Correctly Classified Instances

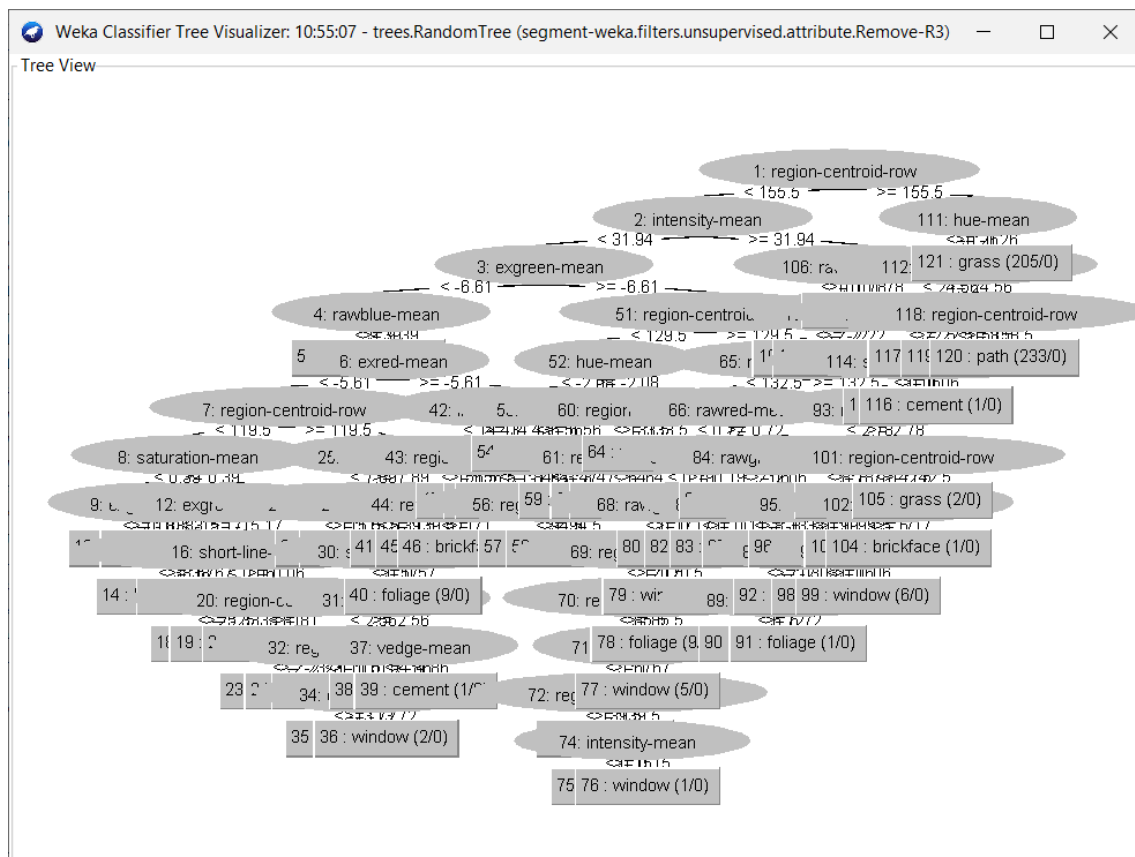
3.1.2 Limitando la profundidad del árbol a 3:

De las 300 instancias clasifica bien 252 => 84 % Correctly Classified Instances. Está claro que necesita más profundidad

3.1.3 Limitando la profundidad del árbol a 5:

De las 300 instancias clasifica bien 271 => 90.33 % Correctly Classified Instances.

El mayor Random Tree es el que se obtiene con los valores por defecto. Tiene 121 nodos y una profundidad máxima de 13. La imagen del modelo no se entiende bien con una imagen por la cantidad de nodos, pero se puede ver en el fichero weka-best-random-tree.txt



3.2 Random Forest

3.2.1 Con todos los valores por defecto:

De las 300 instancias clasifica bien 296 => 98.67 % Correctly Classified Instances

3.2.2 Cambiando las iteraciones a 1000 (por defecto son 100):

De las 300 instancias clasifica bien 298 => 99.33 % Correctly Classified Instances.

3.2.3 Cambiando las iteraciones a 10000 (por defecto son 100):

De las 300 instancias clasifica bien 298 => 99.33 % Correctly Classified Instances. Es mejor el modelo anterior porque este es computacionalmente mucho más exigente (21 segundos para construir el modelo vs 2.1 segundos en el caso anterior)

3.2.4 Cambiando la profundidad máxima a 5 y las iteraciones a 1000 (por defecto son 100):

De las 300 instancias clasifica bien 288 => 96 % Correctly Classified Instances. Es mejor el modelo anterior porque este es computacionalmente mucho más exigente (21 segundos para construir el modelo vs 2.1 segundos en el caso anterior)

El mejor Random Forest es el segundo, con un 99.33% de aciertos en la clasificación (298 de 300). Esta es la matriz de confusión:

=== Confusion Matrix ===

a b c d e f g <-- classified as

41 0 0 0 0 0 0 | a = brickface

0 45 0 0 0 0 0 | b = sky

0 0 40 0 2 0 0 | c = foliage

0 0 0 55 0 0 0 | d = cement

0 0 0 0 34 0 0 | e = window

0 0 0 0 0 45 0 | f = path

0 0 0 0 0 0 38 | g = grass

Solo ha clasificado 2 foliage como si fueran window

se puede ver toda la información del modelo en el fichero weka-best-random-forest.txt

3.3 Logistic

3.3.1 Con todos los valores por defecto:

De las 300 instancias clasifica bien 287 => 95.67 % Correctly Classified Instances

3.3.2 Cambiando maxIts a 100 (por defecto -1):

De las 300 instancias clasifica bien 283 => 94.33 % Correctly Classified Instances.

3.3.3 Cambiando maxIts a 1000 (por defecto -1):

De las 300 instancias clasifica bien 289 => 96.33 % Correctly Classified Instances.

3.3.4 Cambiando maxIts a 10000 (por defecto -1):

De las 300 instancias clasifica bien 289 => 96.33 % Correctly Classified Instances. Es mejor el modelo anterior porque este obtiene el mismo resultado però cuesta más calcularlo.

El mejor Logisitc es el tercero, con un 96.33% de aciertos en la clasificación (289 de 300). Esta es la matriz de confusión:

=== Confusion Matrix ===

a b c d e f g <-- classified as

41 0 0 0 0 0 0 | a = brickface

0 45 0 0 0 0 0 | b = sky

1 0 39 0 2 0 0 | c = foliage

0 0 2 49 3 1 0 | d = cement

1 0 1 0 32 0 0 | e = window

0 0 0 0 0 45 0 | f = path

0 0 0 0 0 38 | g = grass

se puede ver toda la información del modelo en el fichero weka-best-logistic.txt

3.4 Multilayer Perceptron

3.4.1 Con todos los valores por defecto:

De las 300 instancias clasifica bien 293 => 97.67 % Correctly Classified Instances

3.4.2 Cambiando a 2 hidden layers de 15 y 11 neuronas (por defecto "a" que significa automático):

De las 300 instancias clasifica bien 292 => 97.33 % Correctly Classified Instances.

3.4.3 Cambiando a 3 hidden layers de 15, 11 y 9 neuronas (por defecto "a" que significa automático):

De las 300 instancias clasifica bien 282 => 94 % Correctly Classified Instances.

3.4.4 Cambiando a 2 hidden layers de 13 y 9 neuronas (por defecto "a" que significa automático):

De las 300 instancias clasifica bien 290 => 96.67 % Correctly Classified Instances.

El mejor Multilayer Perceptron es el es el primero y el cuarto, con un 96.67% de aciertos en la clasificación (290 de 300). Escojo el último porque conozco la red neuronal utilizada.

Esta es la matriz de confusión:

=== Confusion Matrix ===

a b c d e f g <-- classified as

41 0 0 0 0 0 0 | a = brickface

0 45 0 0 0 0 0 | b = sky

0 0 36 0 6 0 0 | c = foliage

0 0 0 53 1 1 0 | d = cement

0 0 0 1 33 0 0 | e = window

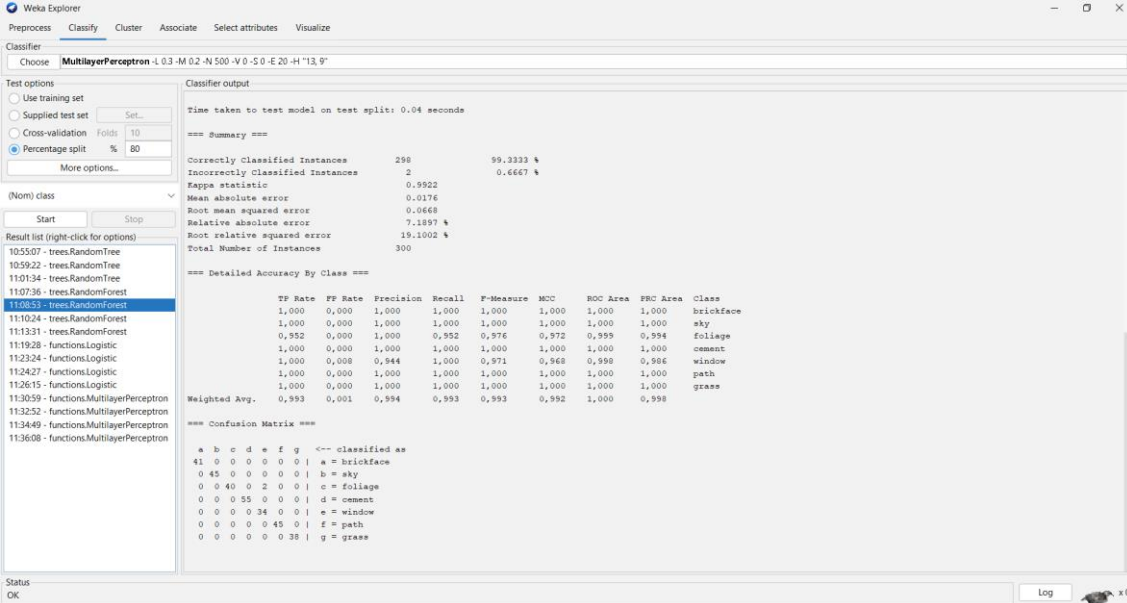
0 0 0 1 0 44 0 | f = path

0 0 0 0 0 0 38 | g = grass

se puede ver toda la información del modelo en el fichero weka-best- weka-best-multilayer-perceptron.txt

4 Elección del mejor modelo

Después de todas las pruebas el mejor modelo lo he conseguido con Random Forest cambiando las iteraciones a 1000. Realiza la clasificación correcta de 298 de las 300 instancias del conjunto de test: 99.33 %



The screenshot shows the Weka Explorer interface with the 'Classify' tab selected. The classifier chosen is 'MultilayerPerceptron' with parameters: -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H "13,9". The test options are set to 'Percentage split' at 80%. The classifier output shows a time taken of 0.04 seconds and a summary of results.

Classifier output

Time taken to test model on test split: 0.04 seconds

=== Summary ===

Metric	Value	Percentage
Correctly Classified Instances	298	99.3333 %
Incorrectly Classified Instances	2	0.6667 %
Kappa statistic	0.9922	
Mean absolute error	0.0176	
Root mean squared error	0.0668	
Relative absolute error	7.1897 %	
Root relative squared error	19.1002 %	
Total Number of Instances	300	

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	PRC Area	Class
1.000	0.000	1.000	1.000	1.000	1.000	1.000	brickface
1.000	0.000	1.000	1.000	1.000	1.000	1.000	sky
0.952	0.000	1.000	0.952	0.976	0.972	0.999	foliage
1.000	0.000	1.000	1.000	1.000	1.000	1.000	cement
1.000	0.008	0.944	1.000	0.971	0.968	0.986	window
1.000	0.000	1.000	1.000	1.000	1.000	1.000	path
1.000	0.000	1.000	1.000	1.000	1.000	1.000	grass
Weighted Avg.	0.993	0.991	0.994	0.993	0.993	0.992	

=== Confusion Matrix ===

a	b	c	d	e	f	g	<-- Classified as
41	0	0	0	0	0	0	a = brickface
0	45	0	0	0	0	0	b = sky
0	0	40	0	2	0	0	c = foliage
0	0	0	59	0	0	0	d = cement
0	0	0	0	34	0	0	e = window
0	0	0	0	0	45	0	f = path
0	0	0	0	0	0	38	g = grass

Status: OK