

Universidad Blas Pascal
Materia: Sistemas de Inteligencia Artificial



Sistemas de Inteligencia Artificial

Trabajo Practico 1

PROFESOR: Agustin Fernandez

ESTUDIANTE:

Antares Jezabel Rosso

Juana Ruiz Luque

Córdoba, 3 de Octubre 2025

Ej.1) (10p)

Dada la siguiente matriz de confusión:

	0	1
0	855	150
1	415	180

Calcular, explicar y documentar los pasos de los cálculos realizados para:

- Exactitud o Accuracy
- TPR
- FPR
- F-measure

	0	1
0	855	150
1	415	180

Exactitud o Accuracy

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FN} + \text{FP})$$

0.646875

True Positive Rate o Recall

$$\text{TPR} = \text{TP} / (\text{TP} + \text{FN})$$

0.8507462687

False Positive Rate

$$\text{FPR} = \text{FP} / (\text{FP} + \text{TN})$$

0.6974789916

Presicion

$$\text{Presicion} = \text{TP} / (\text{TP} + \text{FP})$$

0.6732283465

F-measure

$$\text{F-measure} = 2 * \text{presicion} * \text{recall} / (\text{presicion} + \text{recall})$$

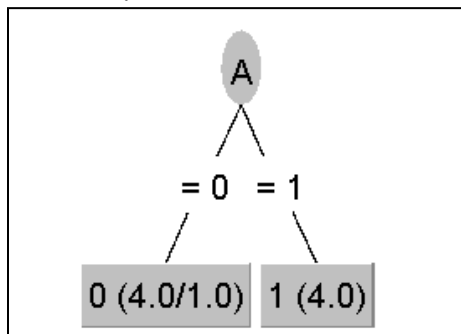
0.7516483516

Ej. 2) (20p)

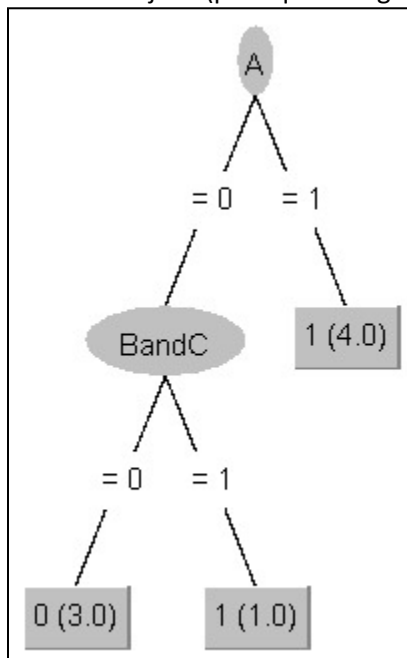
Construya una tabla de decisión para la función booleana A or (B and C) y luego el árbol TDIDT correspondiente. Finalmente construya las reglas de decisión partiendo del árbol de decisión obtenido anteriormente.

A	B	C	B and C	A or (B and C)
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

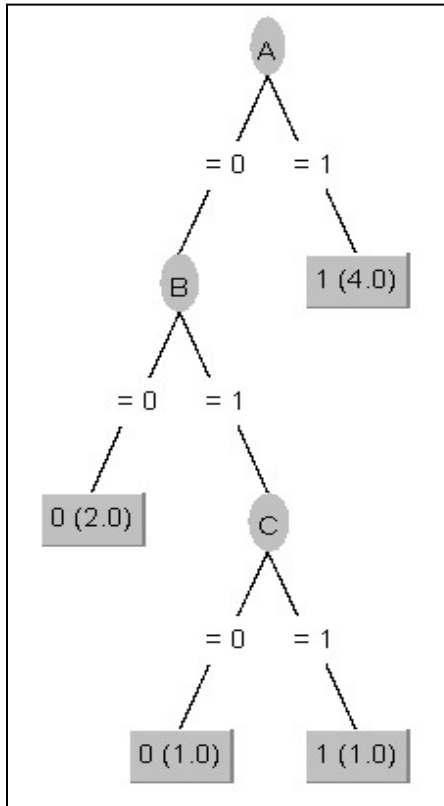
Árbol con poda:



Árbol sin podar: Unpruned = True (esta es la opción clave).
MinNumObj = 1 (para que no agrupe de más).



Árbol con las variables originales A, B y C, para eso eliminamos BandC



Reglas de decicion

```
if (A == 1)
  class = 1
else
  if (B == 0)
    class = 0
  else
    if (c == 0)
      class = 0
    else
      class = 1
```

Ej. 3) (10p)

Dado el siguiente dataset:

A1	A2	Clase
T	T	P
T	T	P
T	F	N
F	F	P
F	T	N
F	T	N

Calcular y documentar los pasos de los cálculos realizados para:

- La información total proporcionada por la tabla.
- La ganancia y entropía de cada atributo.

1. Entropía Inicial (sobre la clase): aplicamos la fórmula general de la entropía.

$$I(P(V_1), \dots, P(V_n)) = - \sum_{i=1}^n P(V_i) \log_2 (P(V_i))$$

$$I(P(p), P(n)) = (-3/6 * \text{LOG}(3/6, 2)) + (-3/6 * \text{LOG}(3/6, 2))$$

1

La entropía inicial es 1 bit

2. Entropía condicional de los atributos.

$$I(P(V_1), \dots, P(V_n)) = - \sum_{i=1}^n P(V_i) \log_2 (P(V_i))$$

Para A1

$$\text{Ev. T: } I(P(p), P(n)) = (-2/3 * \text{LOG}(2/3, 2)) + (-1/3 * \text{LOG}(1/3, 2))$$

0.9182958341

$$\text{Ev. F: } I(P(p), P(n)) = (-1/3 * \text{LOG}(1/3, 2)) + (-2/3 * \text{LOG}(2/3, 2))$$

0.9182958341

Para A2

$$\text{Ev. T: } I(P(p), P(n)) = (-2/4 * \text{LOG}(2/4, 2)) + (-2/4 * \text{LOG}(2/4, 2))$$

1

$$\text{Ev. F: } I(P(p), P(n)) = (-1/2 * \text{LOG}(1/2, 2)) + (-1/2 * \text{LOG}(1/2, 2))$$

1

Aplicamos formula entropia condicional

$$E(A) = \sum_{i=1}^N \frac{p_i + n_i}{P + N} \cdot I(p_i, n_i)$$

E(A1) = (2+1)/6 * 0.9182958341 + (1+2)/6 * 0.9182958341	
	0.9182958341
E(A2) = ((2+2)/6 * 1) + ((1+1)/6 * 1)	
	1

Ganancia de Información

$$G(A) = H_{inicial} - E(A)$$

G(A1)= 1 - 0.9182958341	
	0.0817041659
G(A2)= 1 - 1	
	0

A partir de los cálculos realizados observamos que la ganancia de información del atributo A1 es de 0.081, mientras que la del atributo A2 es de 0. Esto significa que el atributo A1, aunque con un aporte reducido, brinda más información que el atributo A2, el cual no contribuye en absoluto a la clasificación.

Ej. 4) (30p)

Tome los datasets:

- hand_train.arff
- hand_test.arff
- hand externals_test.arff

Construya los flujos de trabajo correspondientes para poder:

- Lleve adelante los cambios necesarios para pre-tratar los datos (si es que es necesario) y que el conjunto de datos quede dispuesto para poder aplicar un algoritmo de clasificación. Documente cuales fueron esos cambios.
- Entrenar un árbol J48 utilizando hand_train.arff y para testear a dicho árbol utilice hand_test.arff.
- Guarde tanto el modelo como los resultados obtenidos.
- Probar el modelo entrenado en el paso anterior utilizando hand_externals_test.arff y finalmente guarde los resultados en un archivo para poder realizar comparaciones con el paso anterior.
- Con el modelo entrenado y elegido prediga cual es la clase (gesto) que se corresponde para el siguiente registro:
[?,0,0,1,15,30,16,29,66,29,26,98,38,6,116,38,20,102,4,21,130,23,20,107,29,15,94,29,0,100,0,1,128,18,4,95,20,0,86,15,-20,93,0,-16,108,18,-9,79,13,-13,72,3,-41,83,0,-30,98,9,-21,78,4,-23,71,-4]

- a) Usando el filtro RemoveUseless y ReplaceMissingValues
b)

Classifier output									
Root relative squared error				3.5441 %					
Total Number of Instances				1792					
=== Detailed Accuracy By Class ===									
	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
1,000	0,001	0,995	1,000	1,000	0,998	0,997	0,999	0,995	s
1,000	0,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1
1,000	0,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	2
1,000	0,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	3
1,000	0,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	4
1,000	0,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	5
1,000	0,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	9
1,000	0,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	L
0,995	0,000	1,000	0,995	0,997	0,997	0,997	0,997	0,996	a
Weighted Avg.	0,999	0,000	0,999	0,999	0,999	0,999	1,000	0,999	
=== Confusion Matrix ===									
a	b	c	d	e	f	g	h	i	<-- classified as
200	0	0	0	0	0	0	0	0	a = s
0	200	0	0	0	0	0	0	0	b = 1
0	0	200	0	0	0	0	0	0	c = 2
0	0	0	200	0	0	0	0	0	d = 3
0	0	0	0	199	0	0	0	0	e = 4
0	0	0	0	0	198	0	0	0	f = 5
0	0	0	0	0	0	196	0	0	g = 9
0	0	0	0	0	0	0	199	0	h = L
1	0	0	0	0	0	0	0	199	i = a

d)

Classifier output									
Root relative squared error				57.9117 %					
Total Number of Instances				4500					
=== Detailed Accuracy By Class ===									
	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,934	0,081	0,590	0,934	0,723	0,704	0,900	0,555	s
	0,798	0,000	1,000	0,798	0,888	0,882	0,969	0,884	1
	0,722	0,006	0,935	0,722	0,815	0,803	0,858	0,706	2
	0,960	0,010	0,923	0,960	0,941	0,934	0,991	0,903	3
	0,926	0,052	0,692	0,926	0,792	0,773	0,937	0,649	4
	0,980	0,008	0,942	0,980	0,961	0,956	0,986	0,926	5
	0,884	0,000	1,000	0,884	0,938	0,933	0,942	0,897	9
	0,966	0,001	0,990	0,966	0,978	0,975	0,982	0,960	L
	0,488	0,010	0,859	0,488	0,622	0,618	0,709	0,476	a
Weighted Avg.	0,851	0,019	0,881	0,851	0,851	0,842	0,919	0,773	
=== Confusion Matrix ===									
a	b	c	d	e	f	g	h	i	<-- classified as
467	0	5	1	0	0	0	0	27	a = s
82	399	14	0	0	1	0	4	0	b = 1
0	0	361	2	134	3	0	0	0	c = 2
0	0	2	480	18	0	0	0	0	d = 3
0	0	1	32	463	4	0	0	0	e = 4
0	0	0	0	10	490	0	0	0	f = 5
0	0	1	5	43	9	442	0	0	g = 9
0	0	2	0	1	1	0	483	13	h = L
243	0	0	0	0	12	0	1	244	i = a

e) La clase predicha es

```

=== Predictions on test set ===

inst#      actual  predicted error prediction
    1         1:?         1:s         0.999

```

Ej. 5) (30p)

Tome el dataset diabetes.arff y luego realice lo siguiente:

- a) Lleve adelante los cambios necesarios para pre-tratar los datos (si es que es necesario) y que el conjunto de datos quede dispuesto para poder aplicar un algoritmo de agrupamiento. Documente cuales fueron esos cambios.
- b) Luego construya un flujo de trabajo, con la herramienta de su preferencia, para poder separar en dos grupos la totalidad de los datos utilizando el algoritmo kmeans.
- c) Realice varios tipos de entrenamientos (partición, validación cruzada, etc), realice comparaciones entre las mismas (dejándolas documentadas) y finalmente quédese con la que considere mejor.
- d) Mediante el método del codo determine cual es la cantidad optima grupos para el dataset proporcionado.
- e) Guarde cada modelo entrenado.
- f) Verifique con el modelo entrenado y elegido a que grupo pertenece el siguiente individuo: **[0,179,77,43,510,43.3,0.122,16]**

a) Abrimos archivo diabetes.arff, aplicamos filtros; RemoveUseless, ReplaceMissingValues, Standardize: Aplicamos Standardize para que todos los atributos queden en la misma escala, es decir, centrados en 0 y medidos en desviaciones estándar. De esta manera, evitamos que atributos con valores más grandes (como glucosa o insulina) dominen la distancia euclídea que usa K-means.

weka.gui.GenericObjectEditor

weka.clusterers.SimpleKMeans

About

Cluster data using the k means algorithm.

More

Capabilities

canopyMaxNumCanopiesToHoldInMemory 100

canopyMinimumCanopyDensity 2.0

canopyPeriodicPruningRate 10000

canopyT1 -1.25

canopyT2 -1.0

debug False

displayStdDevs True

distanceFunction Choose EuclideanDistance -R first-l

doNotCheckCapabilities False

dontReplaceMissingValues False

fastDistanceCalc False

initializationMethod Random

maxIterations 500

numClusters 2

numExecutionSlots 1

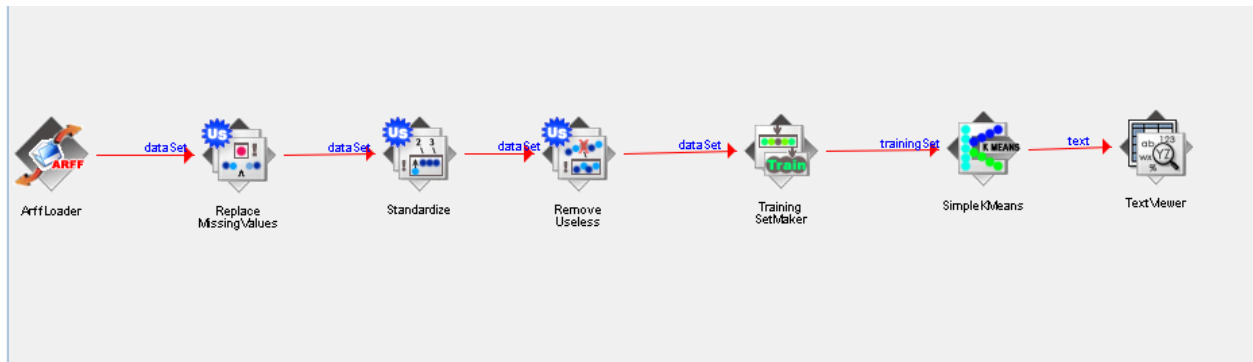
preserveInstancesOrder True

reduceNumberOfDistanceCalcsViaCanopies False

seed 10

Open... Save... OK Cancel

b) Flujo de trabajo



c) Dataset

```

Test
=== Clusterer model ===

Scheme: SimpleKMeans
Relation: pima_diabetes-weka.filters.unsupervised.attribute.ReplaceMissingValues-weka.filters.unsupervised.attribute.Standardize

KMeans
=====

Number of iterations: 3
Within cluster sum of squared errors: 149.51776645811182

Initial starting points (random):

Cluster 0: 0.045984,2.130119,2.112778,0.65593,-0.692439,-0.442987,0.628149,0.319646,tested_negative
Cluster 1: -0.844335,0.691388,0.87284,0.593243,2.17105,-0.240048,1.267997,-0.870806,tested_negative

Missing values globally replaced with mean/mode

Final cluster centroids:

Attribute          Full Data          Cluster#
                   (768.0)            0              1
                   (268.0)            (500.0)
-----
preg              -0                0.3029         -0.1624
plax              0                0.6369         -0.3414
pres             -0                0.0888         -0.0476
skin             -0                0.102          -0.0547
insu             -0                0.1763         -0.0955
mass             0                0.3995         -0.2141
pedi             0                0.2373         -0.1272
age              0                0.3254         -0.1744
class            tested_negative tested_positive tested_negative
  
```

Hold-out

```
Test
=== Clusterer model ===

Scheme: SimpleKMeans
Relation: pima_diabetes-weka.filters.unsupervised.attribute.ReplaceMissingValues-weka.filters.unsupervised.at

kMeans
=====

Number of iterations: 2
Within cluster sum of squared errors: 119.87156111233091

Initial starting points (random):

Cluster 0: -0.844335,-0.966281,-0.367098,-0.535127,-0.319318,-0.607874,0.326334,-0.785774,tested_negative
Cluster 1: -1.141108,1.285646,0.3562,0.969366,-0.692439,2.233265,-0.325587,-0.615709,tested_positive

Missing values globally replaced with mean/mode

Final cluster centroids:

Attribute          Full Data          Cluster#
                   (538.0)          0          1
                   (342.0)          (196.0)
-----
preg               0.0129             -0.1432             0.2852
plas               0.0193             -0.3205             0.6122
pres               0.012              -0.0432             0.1084
skin               -0.0372            -0.1062             0.0831
insu               -0.0095            -0.0731             0.1016
mass               0.0025             -0.2306             0.409
pedi               0.0086             -0.119              0.2313
age                0.0383             -0.148              0.3635
class              tested_negative tested_negative tested_positive
```

Cross-validation

```
Test
=== Clusterer model ===

Scheme: SimpleKMeans
Relation: pima_diabetes-weka.filters.unsupervised.attribute.ReplaceMissingValues-weka.filters.unsupervised.attribute.Standardize-weka.filters.uns

kMeans
=====

Number of iterations: 2
Within cluster sum of squared errors: 161.3151361040585

Initial starting points (random):

Cluster 0: 0.430304,0.410335,0.450535,0.530555,0.450543,0.460516,0.620367,1.785165,tested_positive
Cluster 1: -1.141108,1.285646,0.3562,0.969366,-0.692439,2.233265,-0.325587,-0.615709,tested_negative

Missing values globally replaced with mean/mode

Final cluster centroids:

Attribute          Full Data          Cluster#
                   (481.0)          0          1
                   (335.0)          (464.0)
-----
preg               0.0032             0.3074             -0.146
plas               -0.0125            0.8214             -0.3392
pres               -0.0099            0.0763             -0.0543
skin               0.002              0.1188             -0.2591
insu               0.0051             0.2326             -0.0766
mass               -0.0027            0.3951             -0.2077
pedi               0.007              0.2824             -0.1046
age                -0.0106            0.3008             -0.1712
class              tested_negative tested_positive tested_negative
```

Aplicamos tres formas distintas de particionar los datos para entrenar y evaluar el modelo de agrupamiento no supervisado K-Means:

1. Dataset completo

- Se ejecutó el algoritmo sobre el 100% de las instancias.
- Esto tiene como inconveniente que los mismos datos utilizados para entrenar se emplean también en la evaluación, lo que puede llevar a una sobrestimación del desempeño (no hay datos “nuevos” para probar).
- El **SSE (Sum of Squared Errors)** resultante tiende a ser mayor en valor absoluto porque intervienen más instancias en el cálculo, aunque el centroide obtenido es más representativo al considerar la totalidad de los datos.

2. Hold-Out (70/30)

- El dataset se dividió en un 70% para **entrenamiento** y un 30% para **prueba**.
- En este caso, los centroides se calcularon únicamente con el 70% de los datos, y luego se evaluó con ejemplos distintos (el 30% restante).
- Esto permite estimar la capacidad de generalización del modelo, aunque depende de cómo se haya realizado la partición (puede variar si cambia la división aleatoria).

3. Validación cruzada (10-fold cross validation)

- El dataset se partió en 10 subconjuntos de igual tamaño.
- En cada iteración, 9 partes se usaron para entrenar y la parte restante para probar; el proceso se repitió 10 veces de forma tal que cada instancia fue usada para testear exactamente una vez.
- Este método da una medida de desempeño más estable y confiable, al reducir la dependencia de una única división de los datos.

Observaciones adicionales

- En clustering, como no hay “etiquetas” reales en el entrenamiento, estas técnicas sirven para verificar la consistencia de los grupos obtenidos y comparar la asignación de clusters respecto a la clase real (cuando se dispone de ella, como en el dataset de diabetes).
- El SSE tiende a disminuir en particiones más pequeñas, no porque el modelo sea “mejor”, sino porque hay menos puntos que suman error. Lo importante es comparar valores relativos entre configuraciones (ej: K distinto) más que absolutos.
- En general:
 - Dataset completo → útil para ver la estructura total, pero sobre optimista.
 - Hold-Out → balancea entrenamiento/prueba, más realista.
 - Cross-validation → más robusto y recomendado para evaluar estabilidad del modelo.

d)

K: 2

SSE: 149.51776645811194

K: 3

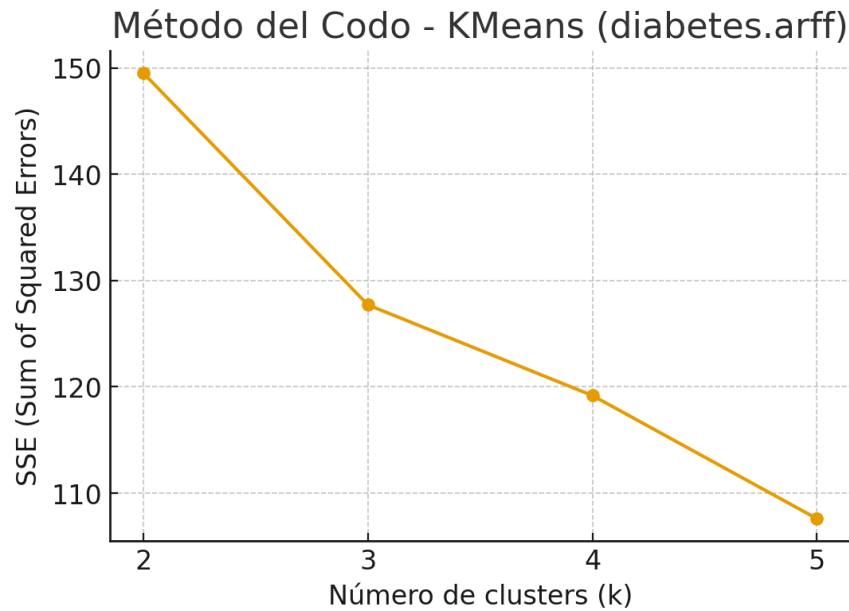
SSE: 127.71162371317592

K: 4

SSE: 119.18257851300015

K: 5

SSE: 107.59456953488935



De acuerdo con el método del codo, el valor óptimo de k es 3, ya que es en este punto donde la curva comienza a aplanarse y la reducción del SSE deja de ser tan pronunciada. Para valores mayores de k (4, 5), la mejora en la compactación de los clusters es marginal en comparación con el salto observado entre $k=2$ y $k=3$. Por lo tanto, $k=3$ representa el número más adecuado de clusters para este dataset.