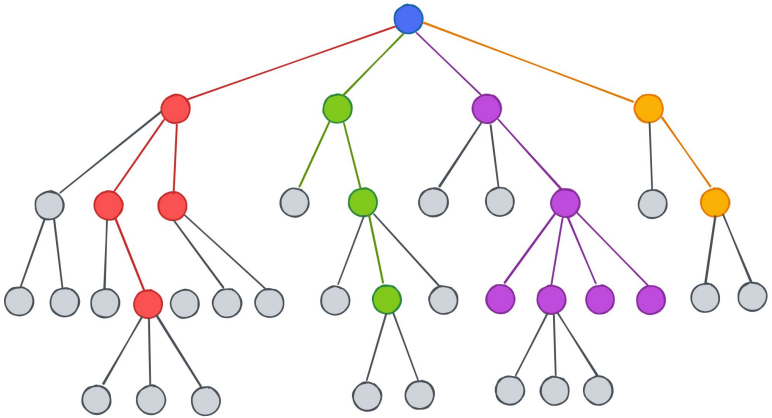
TP2: Algoritmos de Clasificación Supervisada

Dey, Patrick
Lombardi, Matías
Vázquez, Ignacio

Tecnologías utilizadas

Ejercicio 1: Árboles de decisión



Ejercicio 1

- Utilizar árboles de decisión para determinar el otorgamiento de un crédito
- Clase objetivo: Creditability

Preprocesamiento

- Variables: Duration of Credit (month), Credit Amount y Age (years) no son aptas para el árbol
- Las categorizamos de acuerdo a los cuartiles

Duration of Credit (months)	
Rango	Valor asignado
0-12	1
12-18	2
18-24	3
24-72	4

Credit amount	
Rango	Valor asignado
0-1365.5	1
1365.5-2319.5	2
2319.5-3972.25	3
3972.25-18424	4

Age (years)	
Rango	Valor asignado
0-27	1
27-33	2
33-42	3
42-75	4

Implementación: fórmulas previas

Entropía de Shannon

$$H(S) = -\sum_i p_i * log_2(p_i)$$

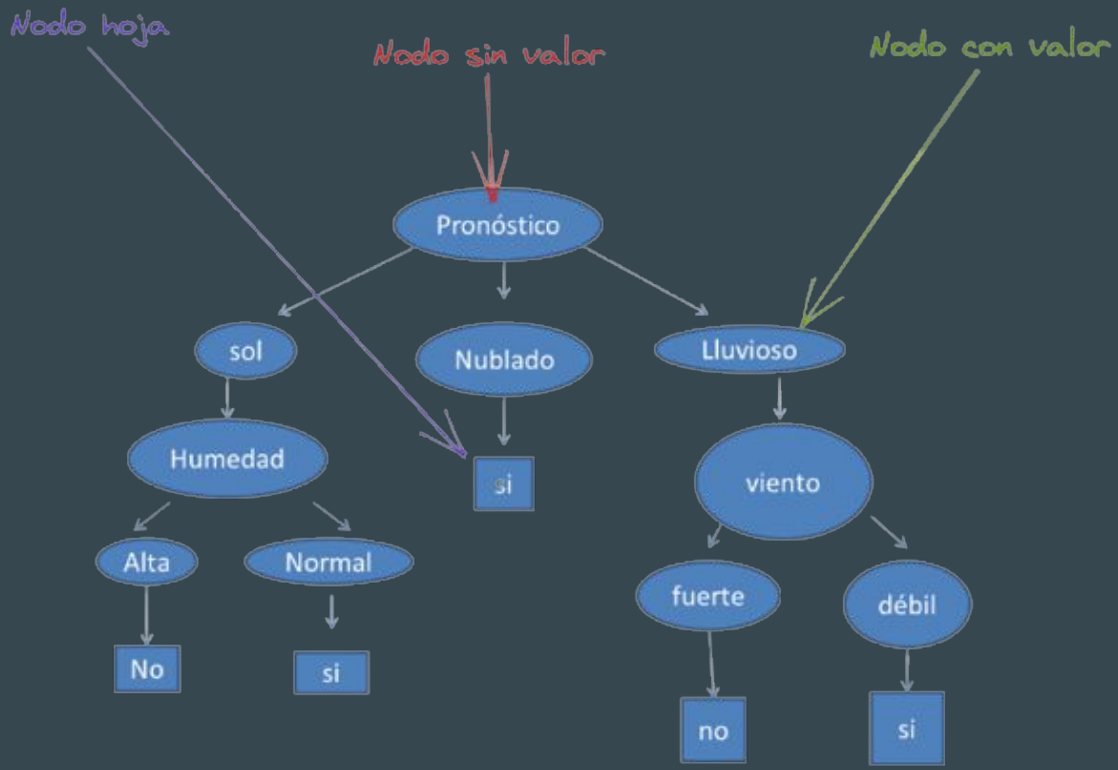
Información de Ganancia

$$H(S, A) = H(S) - \sum_{v \in Atributos(A)} \frac{|S_v|}{|S|} * H(S_v)$$

Implementación: nodos

- Nodos tienen un mapa con sus hijos. Tres tipos de nodos:
  - Sin valor: son los atributos que maximizan la ganancia. Ejemplo: Credit Amount. Tienen múltiples hijos. Cuentan para la máxima cantidad de nodos.
  - Con valor: son todos los posibles valores que puede tomar el atributo que maximiza la ganancia. Ej: [1-4] en Credit Amount. Tienen un solo hijo No cuentan para la máxima cantidad de nodos.
  - Hoja: es la clasificación del árbol. En este caso es binaria. No tienen hijos. Tampoco cuentan para la máxima cantidad de nodos.

Clasificación de nodos



Implementación: construcción del árbol

- Algoritmo ID3 de forma recursiva
- Pasos:
    1. Creación de nodo raíz en base a máxima ganancia. Si hay una sola clase (del atributo objetivo) o no hay atributos: nodo hoja con esa clase.
    2. Ramificación en base a los valores que puede tomar el atributo. Un hijo por cada uno.
    3. Por cada hijo, se repite el mismo análisis que en el primer punto.
    4. Repetir pasos 2 y 3 cortando en las mismas condiciones que en el punto 1.

Implementación: clasificación

- Pasos:
    1. Current = raíz
    2. Mientras current no sea una hoja (es decir, atributo != objetivo)
        a. Si es un nodo sin valor, se busca el hijo que coincida con el atributo
            i. Si existe, current = hijo
            ii. Si no existe (por cómo se construyó el árbol), se retorna la clasificación más probable hasta ese momento
            iii. Si no tiene hijos asociados a sus atributos, quiere decir que es una hoja, por lo que se retorna el valor del hijo (asociado con la post-poda, después se explicará)
        b. Si es un nodo con valor, current = único hijo
    3. Retornar el valor de la hoja

Random Forest

1. Se divide el *dataset* en entrenamiento y prueba
2. De los N ejemplos del conjunto de entrenamiento, se toman m <= N muestras con reemplazo.
3. Con estos ejemplos se construye un árbol
4. Los pasos 2 y 3 se repite k veces
5. La predicción será la clasificación más votada

Métricas

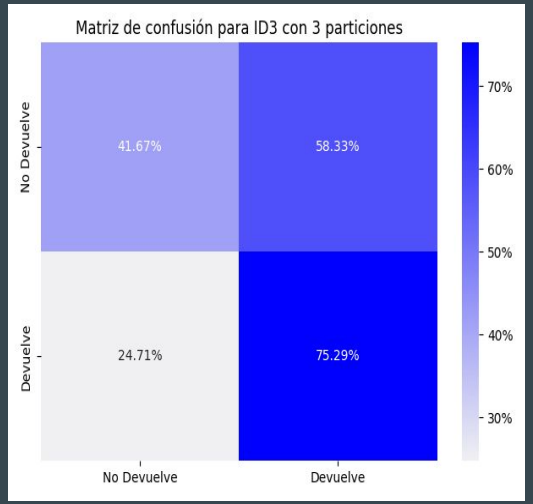
ID3

Métricas: ID3

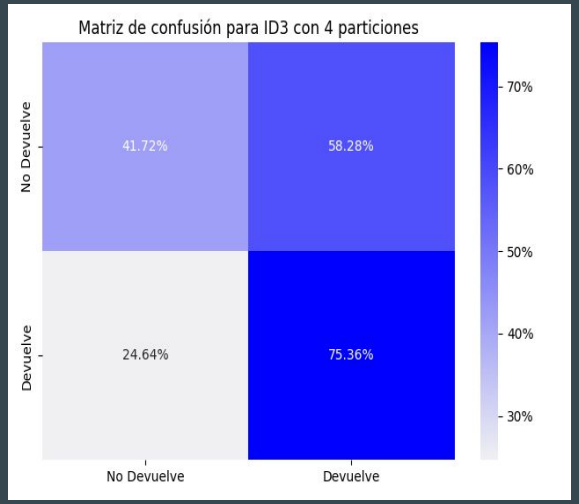
- Shuffle previo del dataset
- Cross-validation, con particiones 70/30, 80/20, 90/10
- Se presenta la precisión promedio y el desvío estándar de todas las corridas
- Para la matriz de confusión, se toma la suma de todas las iteraciones
- No se aplican restricciones sobre el árbol

Matriz de confusión

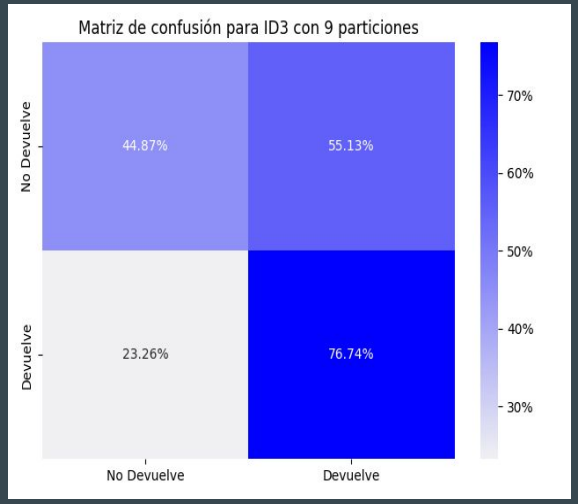
70/30



80/20



90/10



Matriz de confusión

Particiones	Media	Desvío std	Precisión Máxima
70/30	0.65	0.02	0.67
80/20	0.68	0.02	0.71
90/10	0.69	0.02	0.72

Métricas

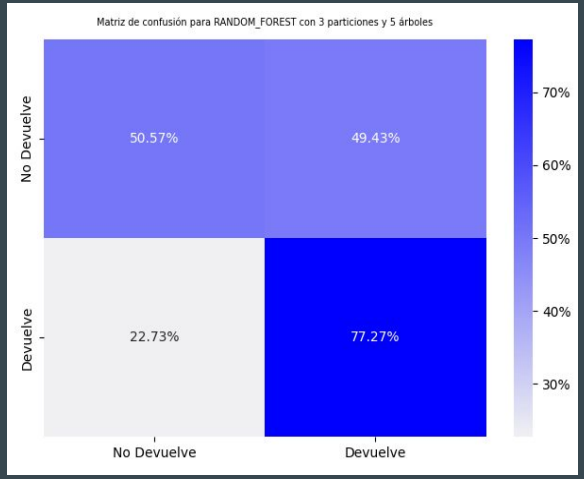
Random Forest

Métricas: Random Forest

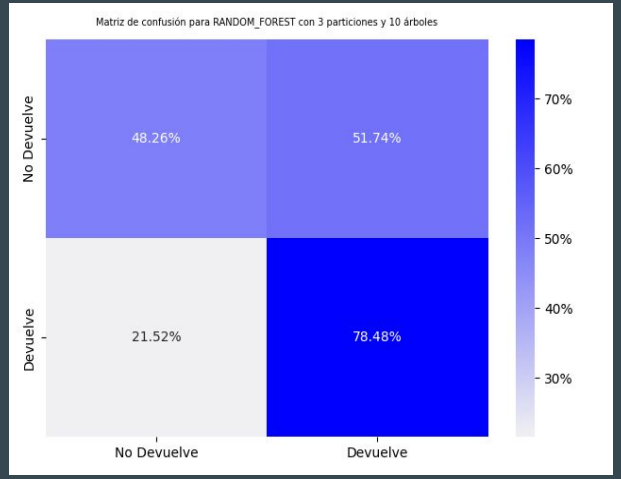
- Shuffle previo del dataset
- Cross-validation, con particiones 70/30, 80/20, 90/10
- Cantidad de árboles : [5, 10, 15]
- Cantidad de ejemplos por árbol es máxima
- Se presenta la precisión promedio y el desvío estándar de todas las corridas
- Para la matriz de confusión, se toma la suma de todas las iteraciones
- No se aplican restricciones sobre el árbol

Resultados 70/30, N° particiones = 3

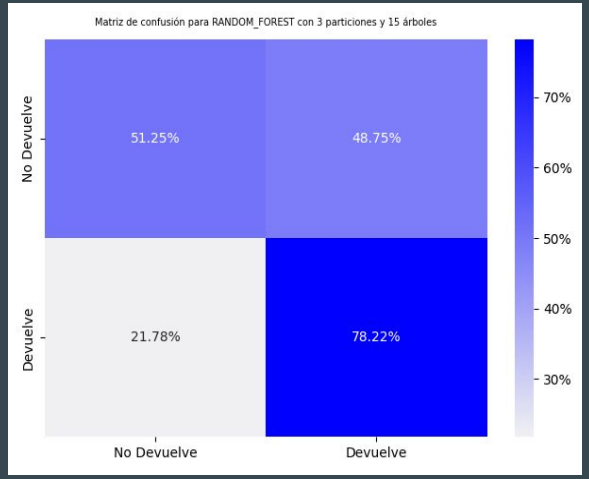
5 Árboles



10 Árboles



15 Árboles

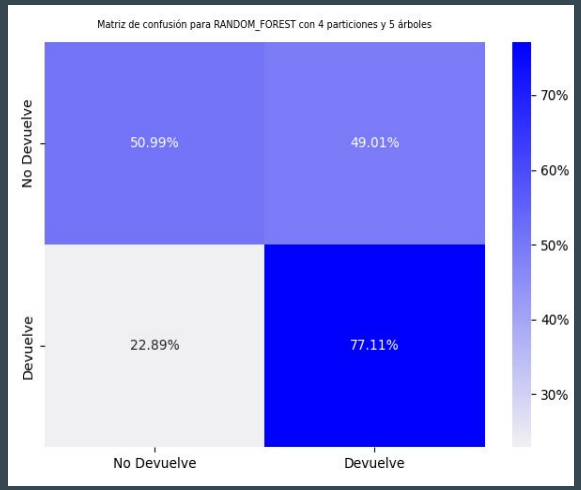


Resultados 70/30, N° particiones = 3

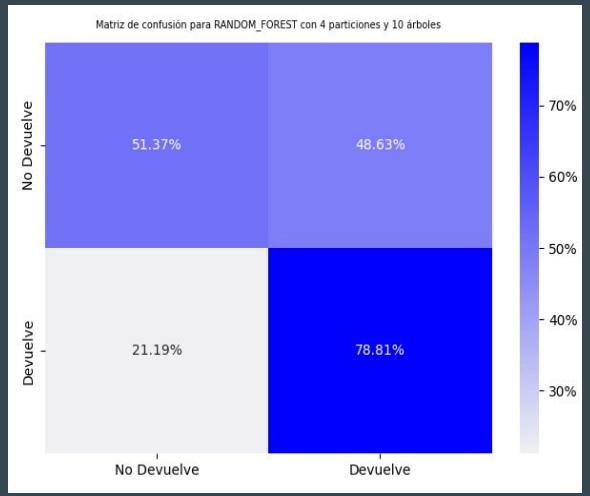
#Árboles	Media	Desvío std	Precisión Máxima
5	0.7	0.02	0.73
10	0.69	0.02	0.7
15	0.71	0.01	0.73

Resultados 80/20, N° particiones = 4

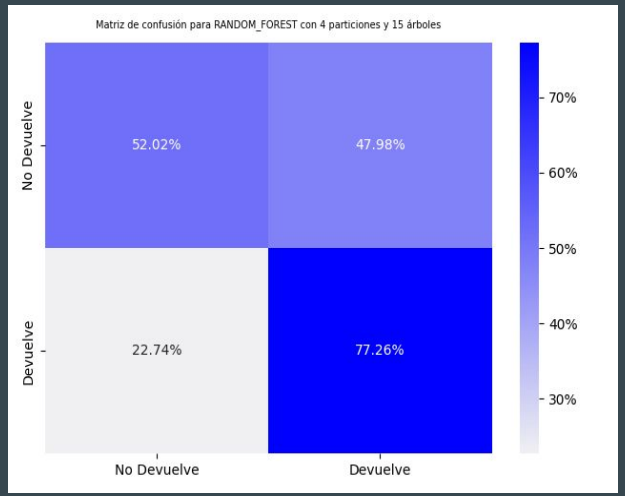
5 Árboles



10 Árboles



15 Árboles

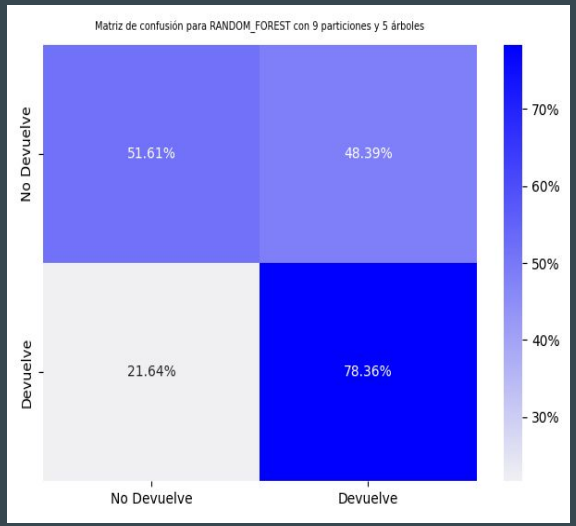


Resultados 80/20, N° particiones = 4

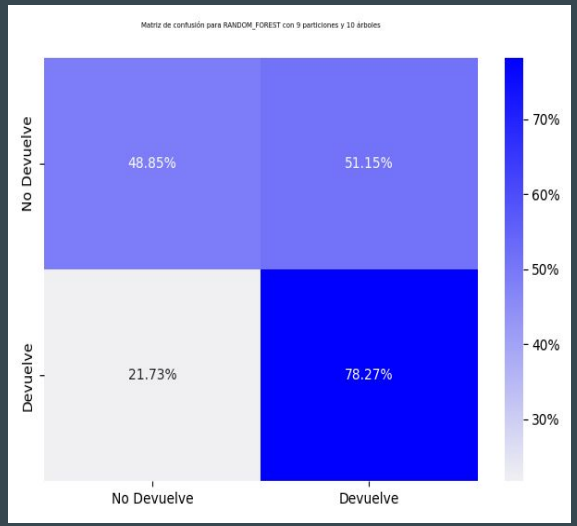
#Árboles	Media	Desvío std	Precisión Máxima
5	0.71	0.02	0.74
10	0.7	0.03	0.74
15	0.71	0.02	0.73

Resultados 90/10, N° particiones = 9

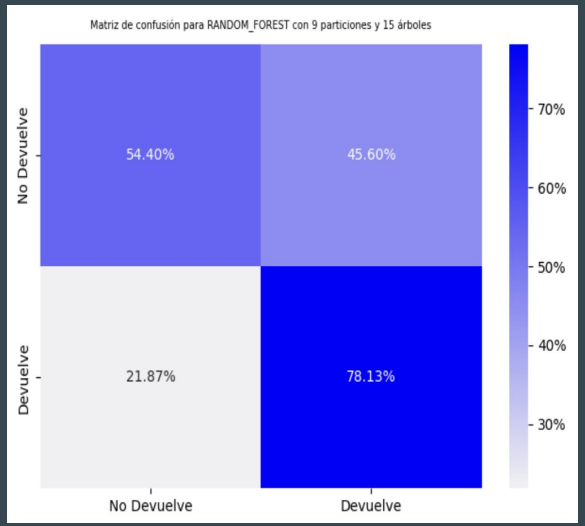
5 Árboles



10 Árboles



15 Árboles



Resultados 90/10, N° particiones = 9

#Árboles	Media	Desvío std	Precisión Máxima
5	0.71	0.03	0.75
10	0.69	0.05	0.77
15	0.72	0.05	0.83

Caso Particular

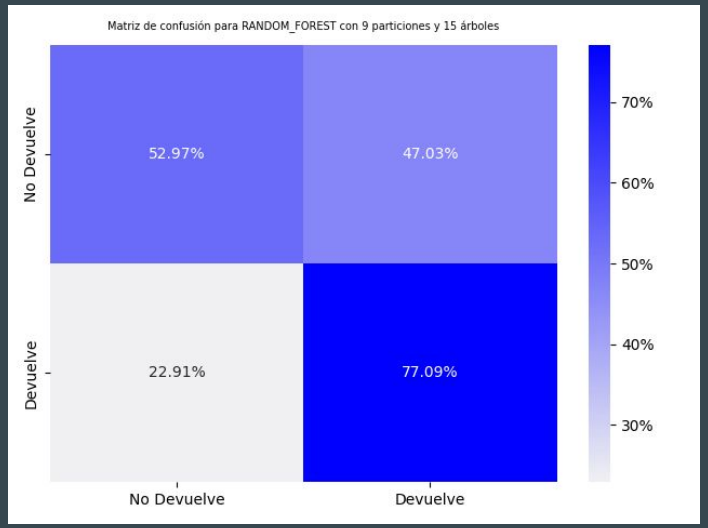
Variando cantidad de ejemplos por árbol en Random Forest

Estudio variando cantidad de ejemplos por árbol

- Partición 90/10
- 15 árboles
- Porcentaje de ejemplos = [0.25, 0.5, 0.75]

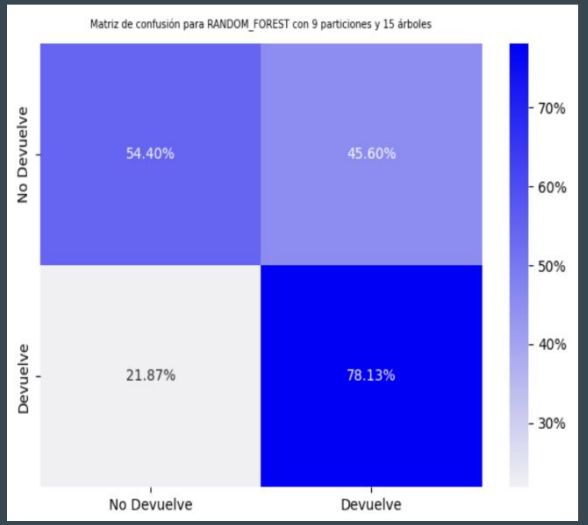
Resultados 25% de cantidad máxima

25%



Media	Desvío std	Precisión Máxima
0.72	0.04	0.8

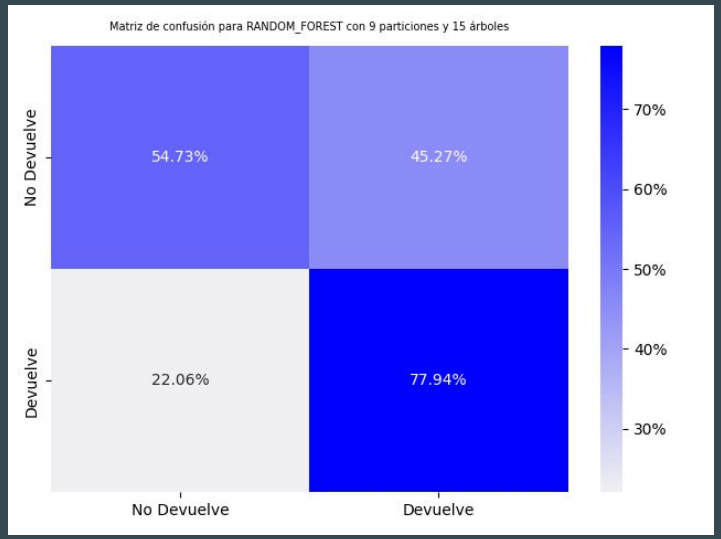
100%



Media	Desvío std	Precisión Máxima
0.72	0.05	0.83

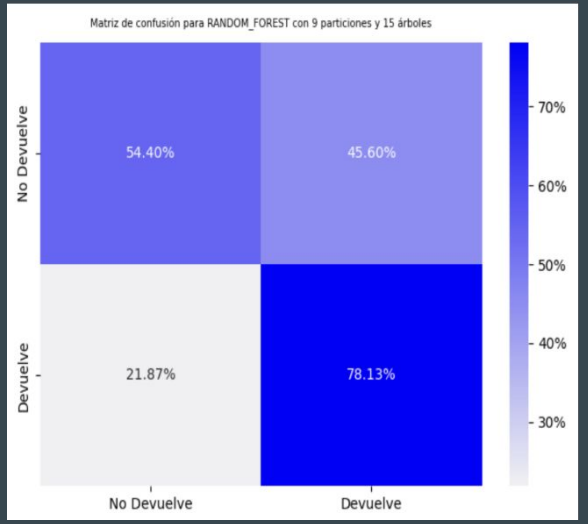
Resultados 50% de cantidad máxima

50%



Media	Desvío std	Precisión Máxima
0.72	0.05	0.78

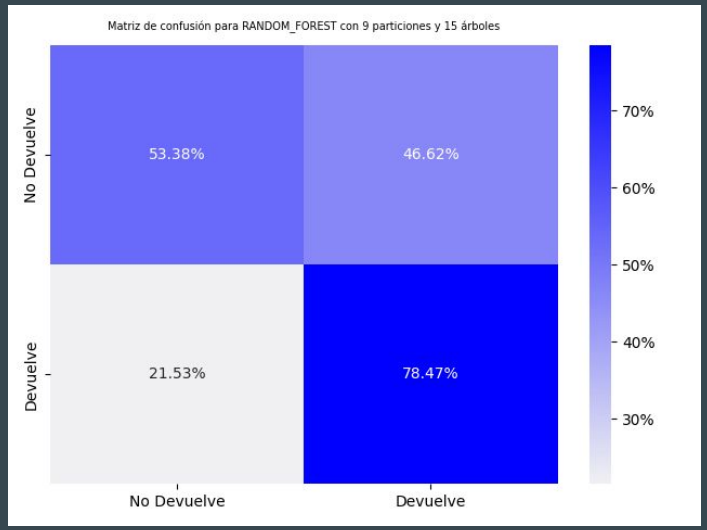
100%



Media	Desvío std	Precisión Máxima
0.72	0.05	0.83

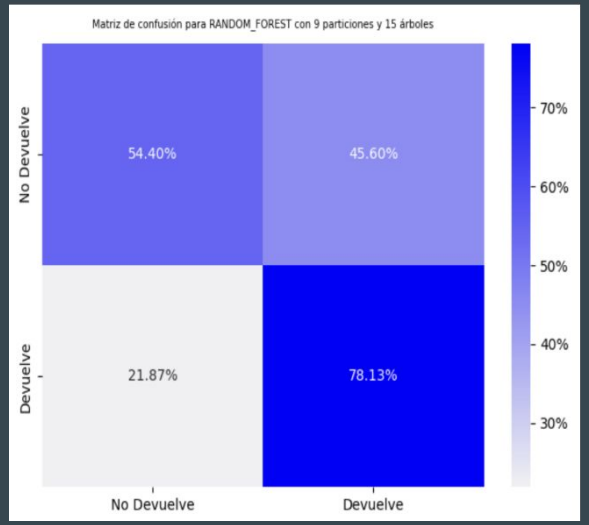
Resultados 75% de cantidad máxima

75%



Media	Desvío std	Precisión Máxima
0.72	0.05	0.82

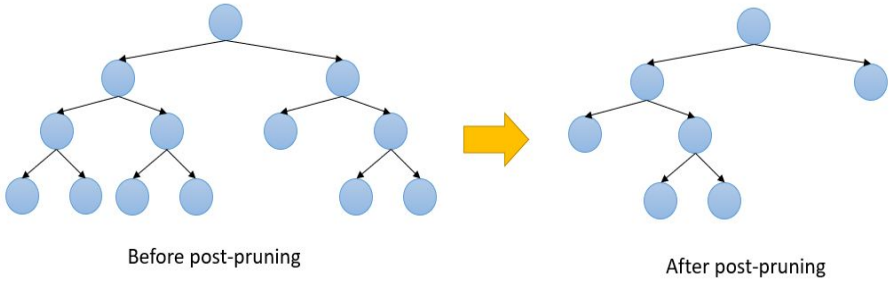
100%



Media	Desvío std	Precisión Máxima
0.72	0.05	0.83

Restricciones

Poda del árbol

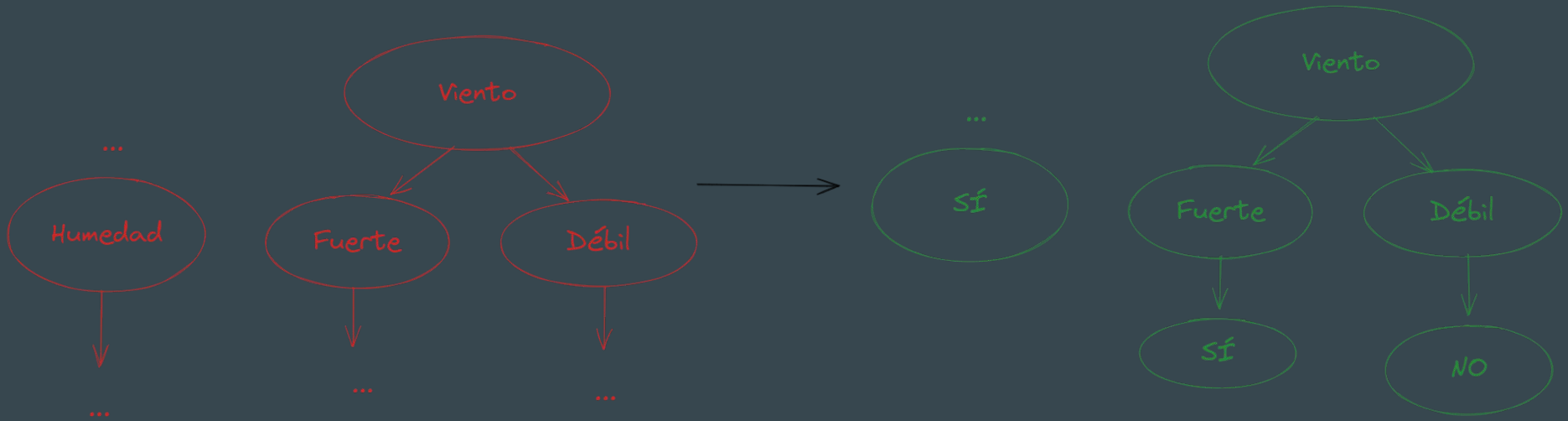


Post-Poda: máxima cantidad de nodos

- Se establece una máxima cantidad de nodos. Solo los nodos sin valor cuentan.
- Se crea el árbol
- Se recorre el árbol de la siguiente manera
  - Nodos = [raíz]
  - Mientras la cantidad de nodos sea menor a la máxima
    - Se cuentan los nodos en Nodos
    - Se crea una lista con los hijos de cada uno
    - Se limpia la lista Nodos y se asignan los hijos del paso anterior
  - Para los nodos que quedaron en Nodos:
    - Si es un nodo sin valor, se sube al padre (no tiene sentido ramificar en un atributo si no se consideran sus valores)
    - Para todos, se crea un hijo que será la clasificación más probable teniendo en cuenta todas las restricciones

Ejemplo clasificación

- Tomemos como ejemplo que Nodos quedó = [Humedad, Fuerte, Débil]

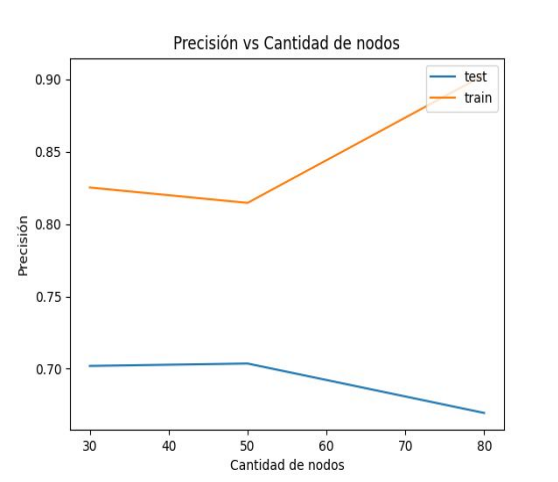


Curvas de precisión: ID3

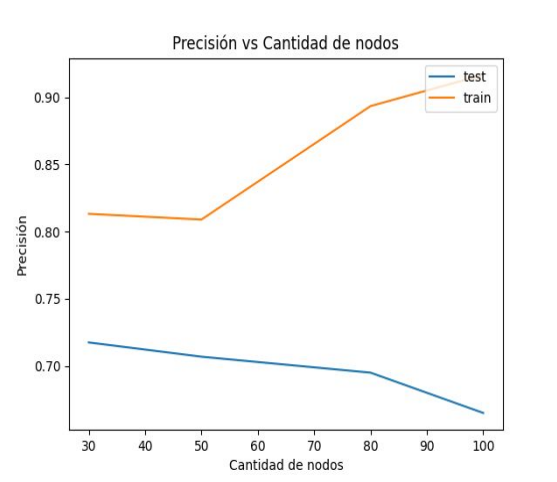
- Nuevamente se utiliza *cross validation*
- Se utiliza máxima cantidad de nodos: [10, 30, 50, 80, 100]
- Por cada cantidad, se poda el árbol como se explicó anteriormente
- Se clasifican los ejemplos como se mencionó previamente
- La precisión es el promedio de todas las particiones

Resultados ID3

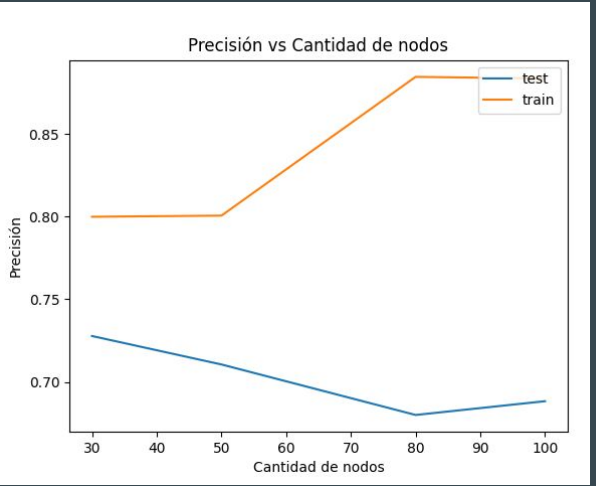
70/30



80/20



90/10

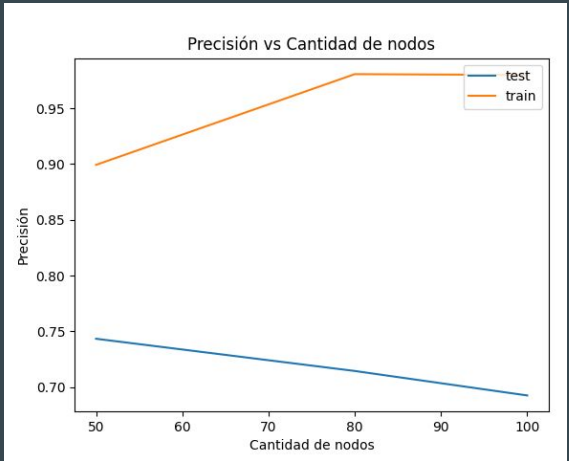


Curvas de precisión: Random Forest

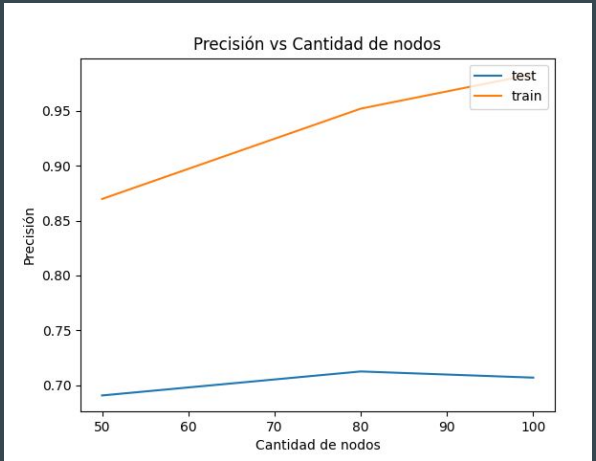
- Nuevamente se utiliza *cross validation*
- Se utiliza máxima cantidad de nodos: [50, 80, 100]
- Se utiliza cantidad de árboles = [10, 15]
- Por cada cantidad, se poda el árbol como se explicó anteriormente
- Se clasifican los ejemplos como se mencionó previamente
- La precisión es el promedio de todas las particiones

Resultados Random Forest 10 árboles

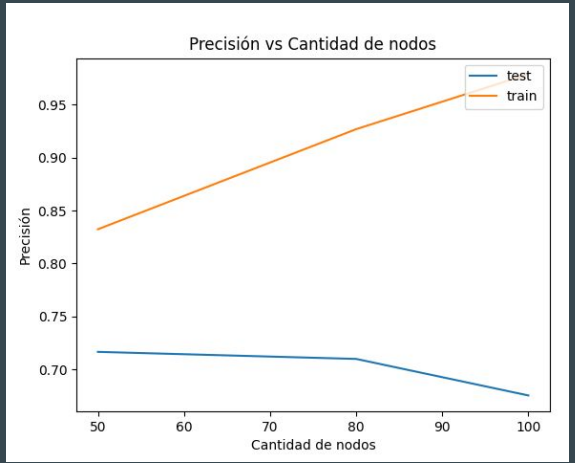
70/30



80/20

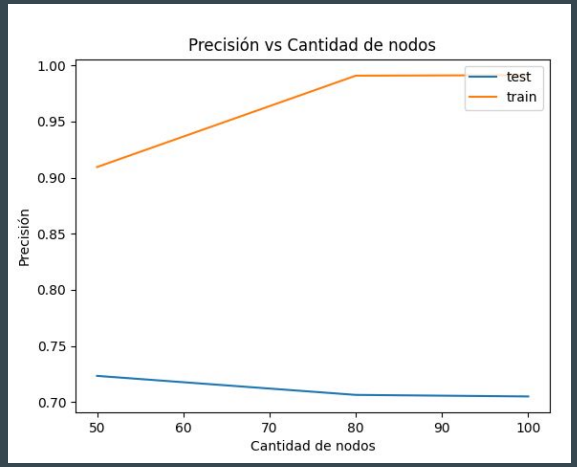


90/10

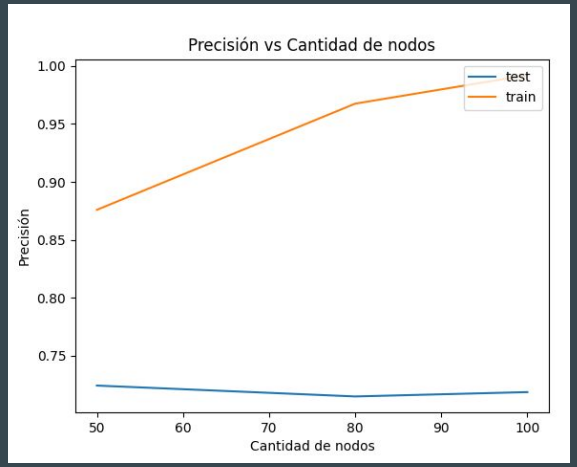


Resultados Random Forest 15 árboles

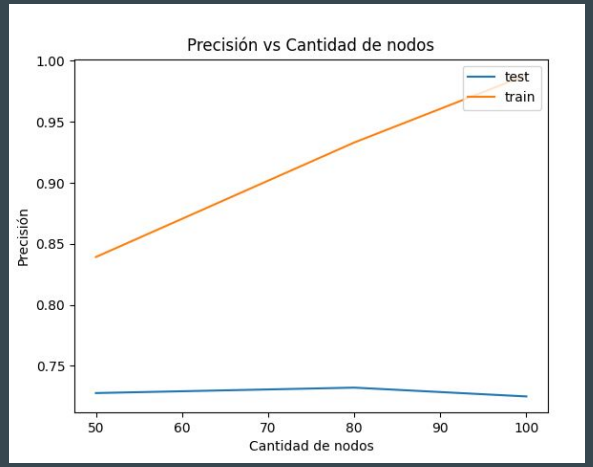
70/30



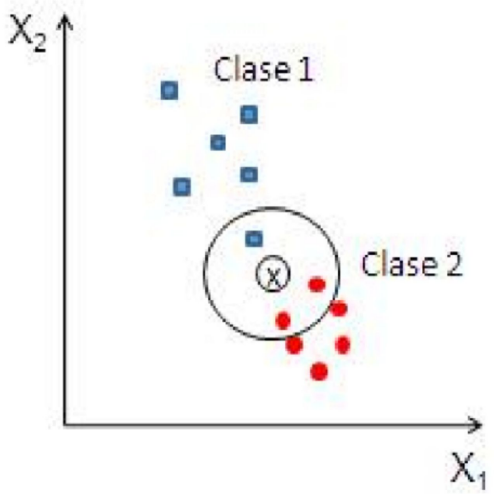
80/20



90/10



Ejercicio 2: KNN y KNN con distancias



Ejercicio 2.a

Cantidad promedio de palabras para los comentarios de 1 estrella

- Sumamos todos los wordcount de todas las instancias cuyo StarRating es 1
- Dividimos por la cantidad de instancias de StarRating = 1

Resultado = 12.216

Ejercicio 2

Clasificar una opinión utilizando los algoritmos K-NN y K-NN con distancias pesadas para distintos valores de k.

- Variable objetivo: Star Rating
- Variables explicativas: wordcount, Title sentiment, sentimentValue

Implementación: preprocesamiento

- Hay instancias que no tienen asignado un valor a titleSentiment.
- Aquellas cuyo StarRating es mayor o igual a 3 se le asigna positive. Sino, negative.
- Normalizamos los datos de entrada salvo la variable StarRating, ya que tienen órdenes de magnitud distintos.
- Dividimos el dataset en conjunto de entrenamiento y de testeo.

Implementación KNN

1. Se almacenan tanto los atributos como la clase de todas las instancias de entrenamiento con el formato: (atributos, clase)
2. Al presentar una nueva instancia se calculan las distancias euclídeas entre los atributos de las instancias almacenadas y la presentada.
3. Tomamos las k instancias a menor distancia y contamos la cantidad de clases.
4. Clasificamos a la instancia presentada con la clase que tiene más apariciones.
5. En caso de empate repetiremos los pasos del 2 al 4 incrementando k en 1 hasta que no lo haya.

Implementación KNN con distancias pesadas

Los pasos son los mismos que en la diapositiva anterior, salvo que se ponderan las distancias en vez de contar únicamente la cantidad de apariciones por clase de la siguiente manera:

$$w_i = \frac{1}{d(x_q, x_i)^2} \qquad \hat{f}(x_q) = arg\,máx_{v \in V} \left(\sum_i W_i * 1_{\{v = f(x_i)\}} \right)$$

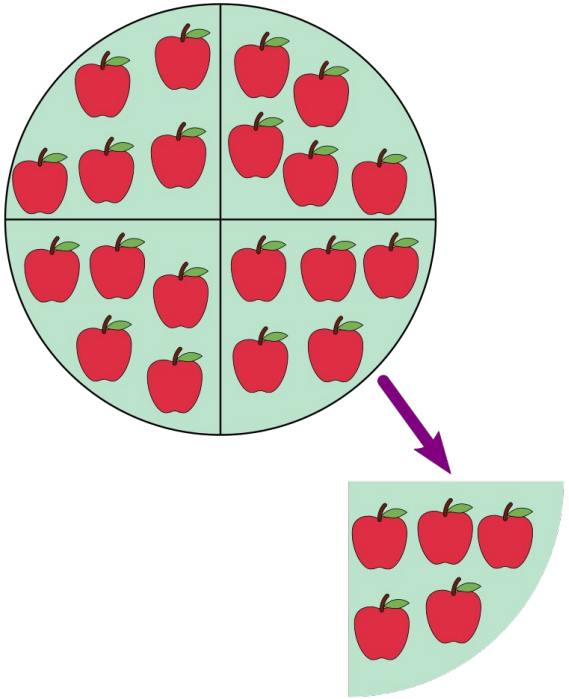
Caso particular: en caso de haber distancia 0, se toma la clase que más distancias 0 tiene.

Métricas para KNN y KNN pesado

- Shuffle previo del dataset
- Cantidad de vecinos k = [5, 8, 10]
- Cross-validation, con particiones 70/30, 80/20, 90/10
- Se presenta la precisión promedio y el desvío estándar de todas las corridas
- Para la matriz de confusión, se toma la suma de todas las iteraciones

Métricas

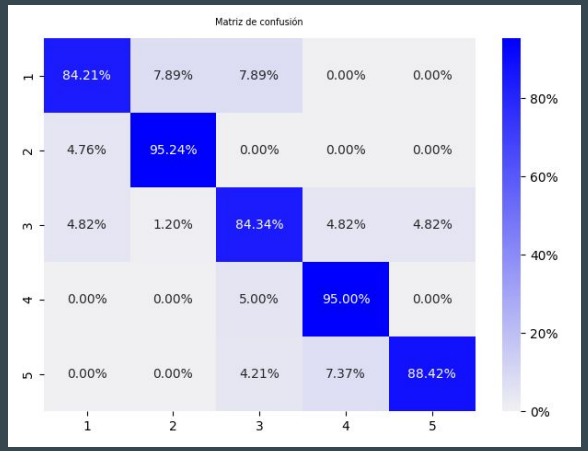
70/30



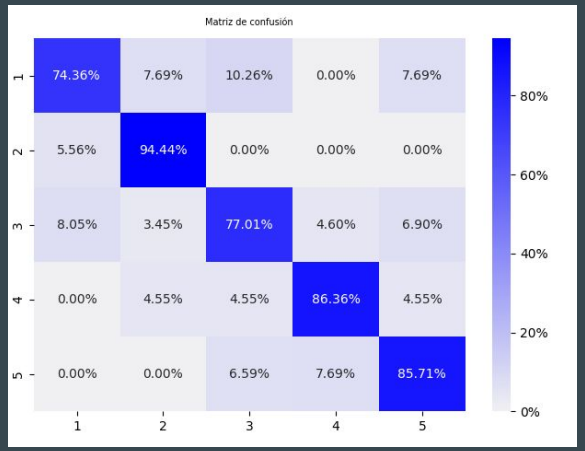
Matriz de confusión

KNN 70/30

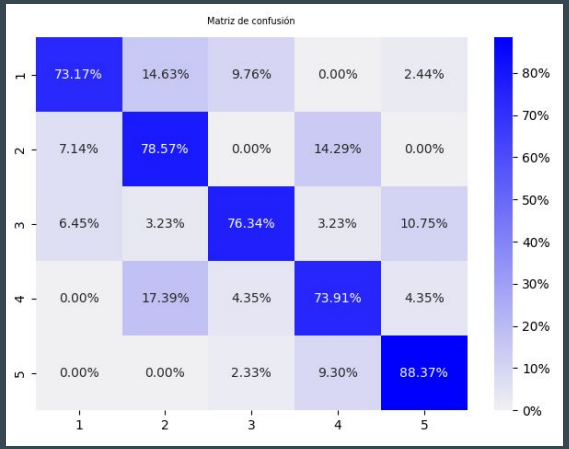
k = 5



k = 8



k=10



Precisión

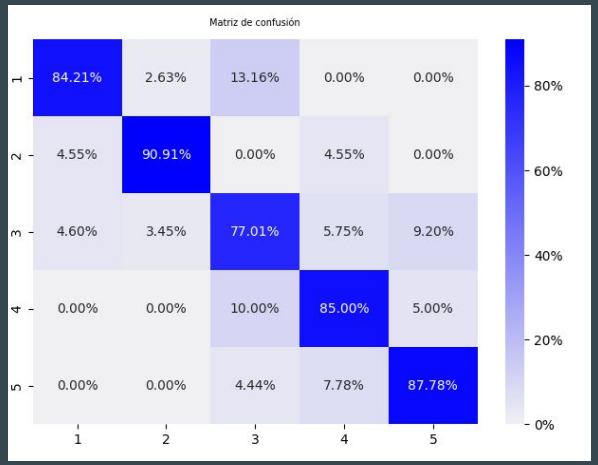
KNN 70/30

	Media	Desvío std
k = 5	0.87	0.02
k = 8	0.81	0.03
k = 10	0.80	0.07

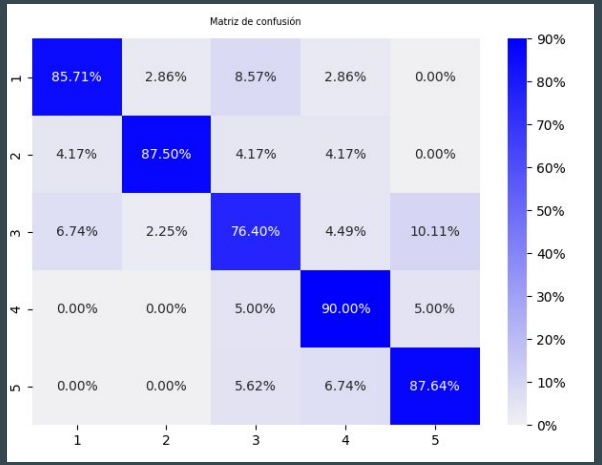
Matriz de confusión con distancias PESADAS

KNN con distancias pesadas 70/30

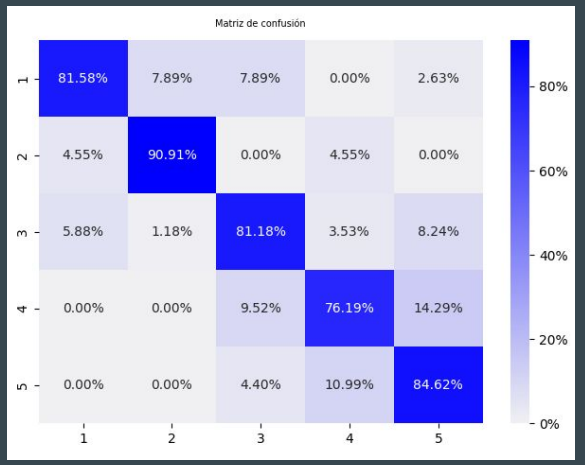
k = 5



k = 8



k=10



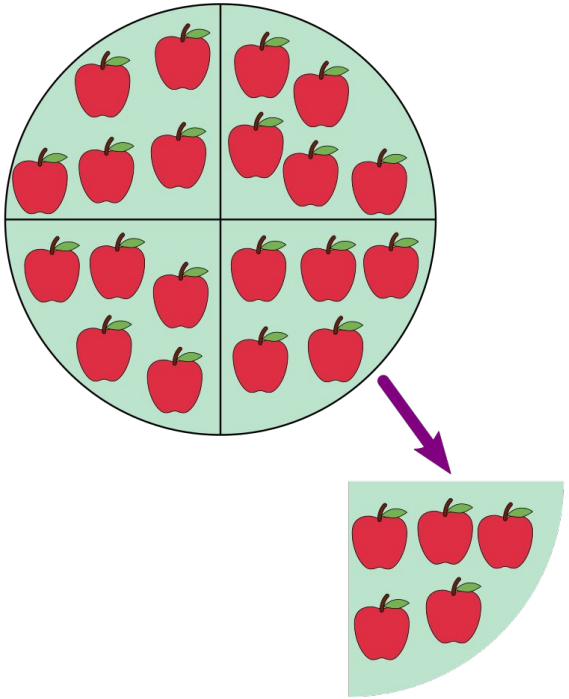
Precisión con distancias PESADAS

KNN con distancias pesadas 70/30

	Media	Desvío std
k = 5	0.84	0.03
k = 8	0.84	0.05
k = 10	0.83	0.04

Métricas

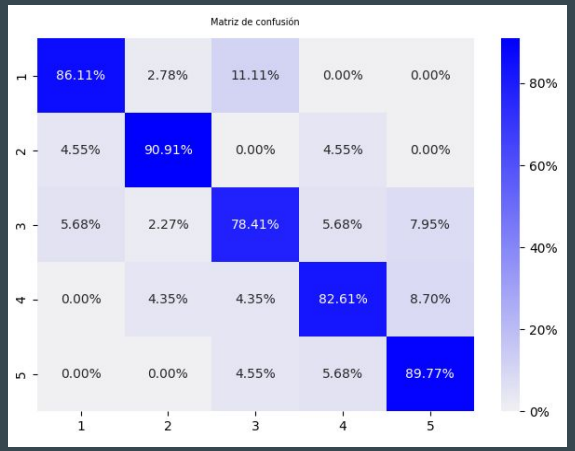
80/20



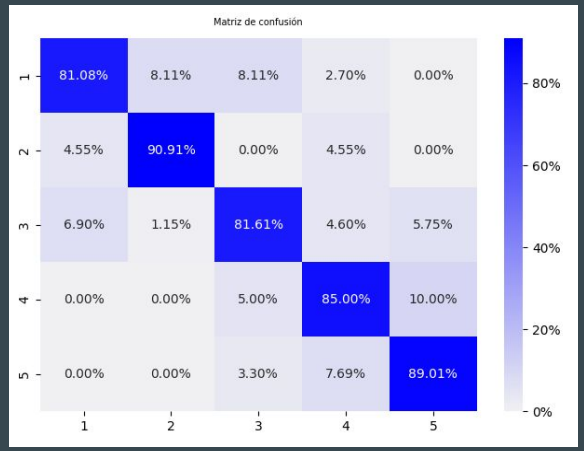
Matriz de confusión

KNN 80/20

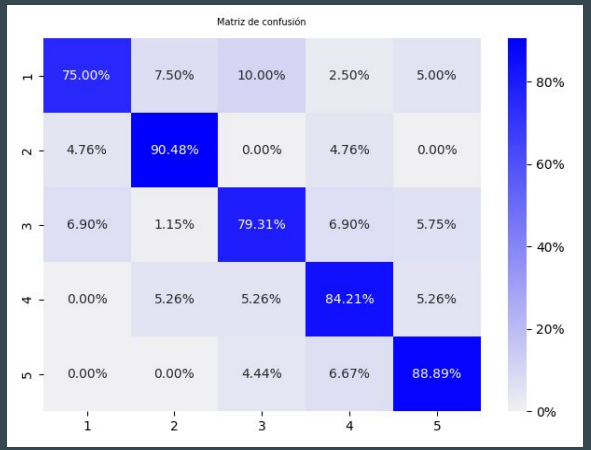
k = 5



k = 8



k=10



Precisión

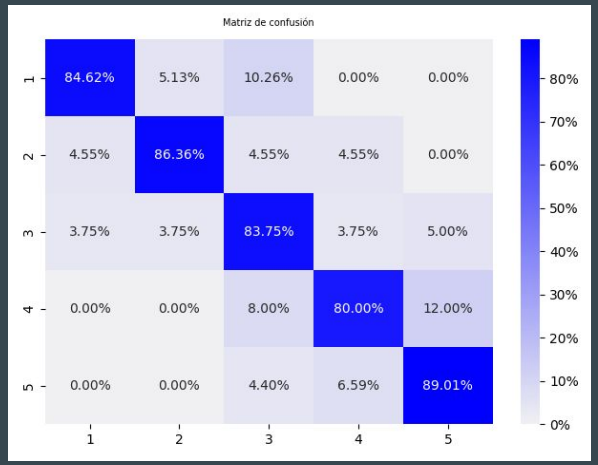
KNN 80/20

	Media	Desvío std
k = 5	0.85	0.06
k = 8	0.85	0.02
k = 10	0.83	0.09

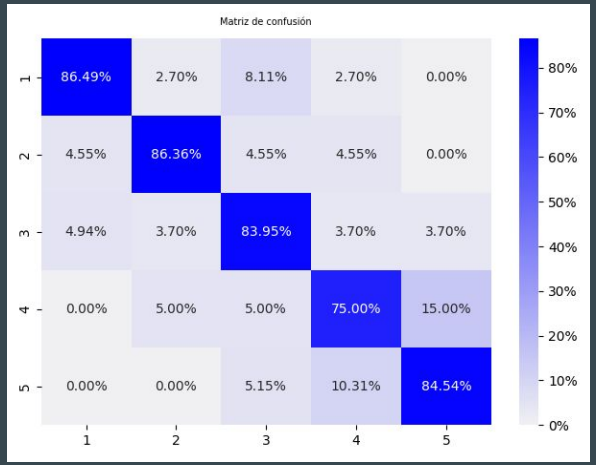
Matriz de confusión con distancias PESADAS

KNN con distancias pesadas 80/20

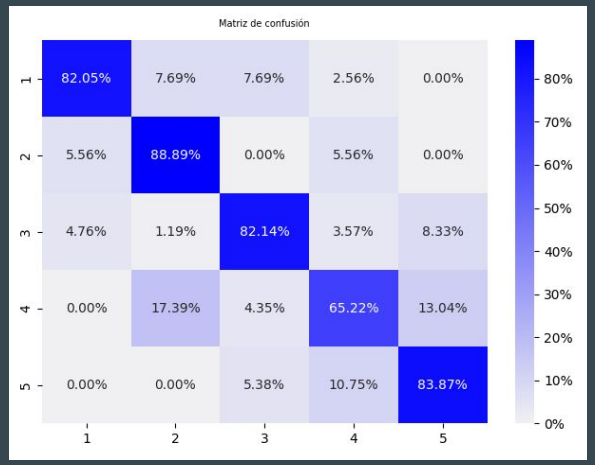
k = 5



k = 8



k=10



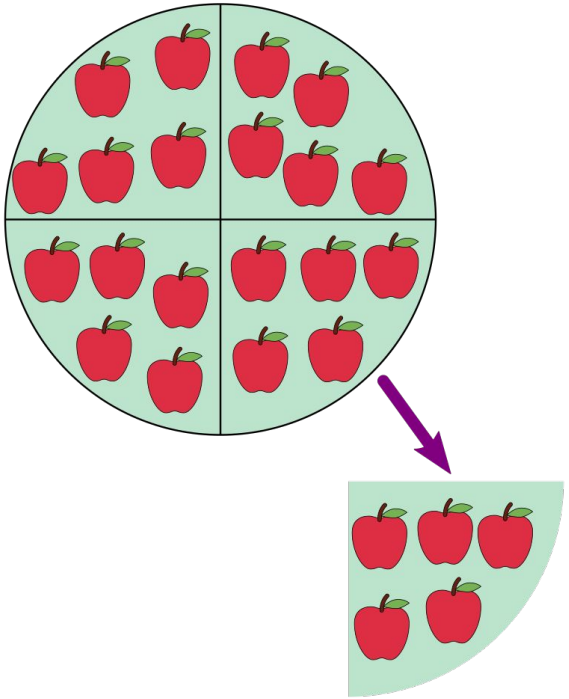
Precisión con distancias PESADAS

KNN con distancias pesadas 80/20

	Media	Desvío std
k = 5	0.86	0.06
k = 8	0.84	0.06
k = 10	0.82	0.03

Métricas

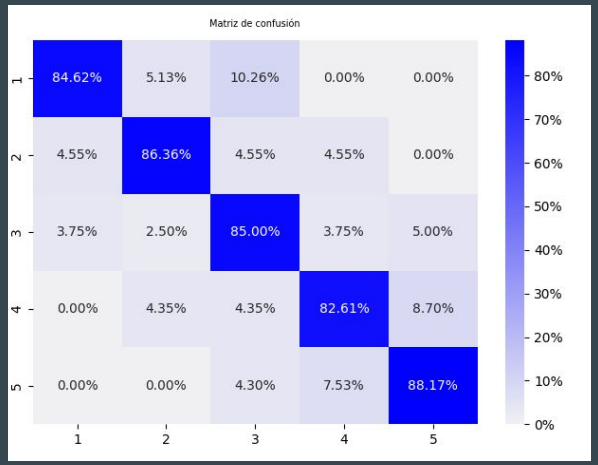
90/10



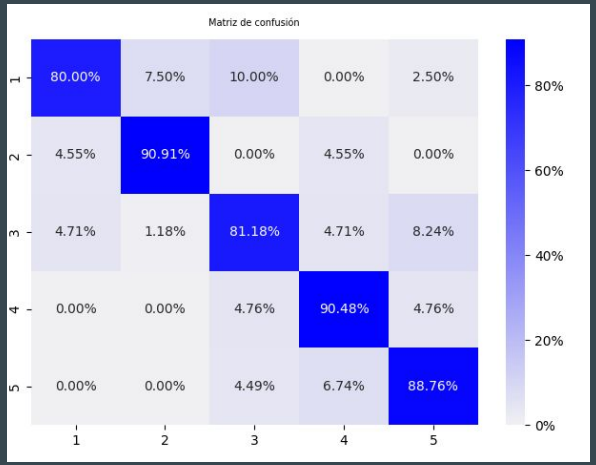
Matriz de confusión

KNN

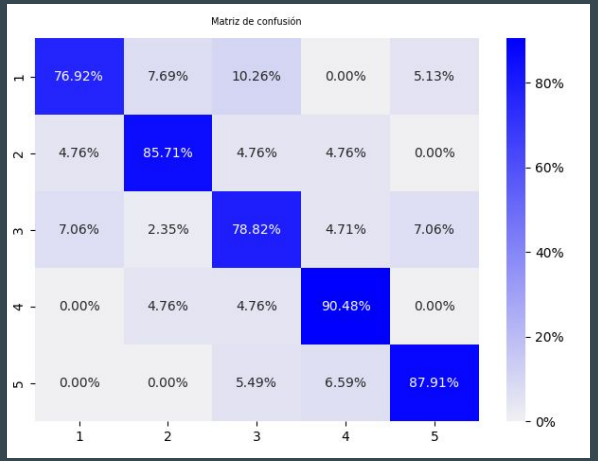
k = 5



k = 8



k=10



Precisión

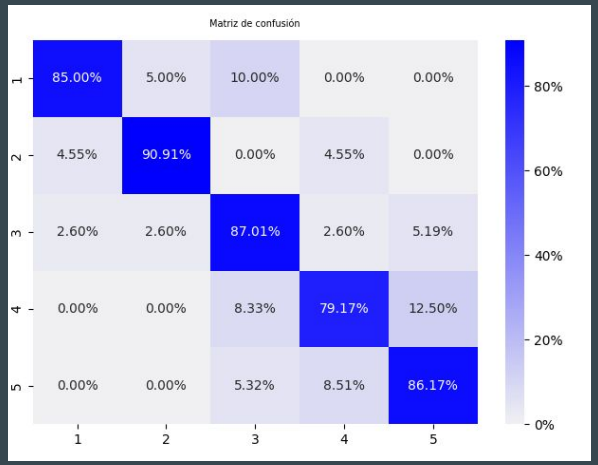
KNN

	Media	Desvío std
k = 5	0.86	0.05
k = 8	0.85	0.06
k = 10	0.83	0.09

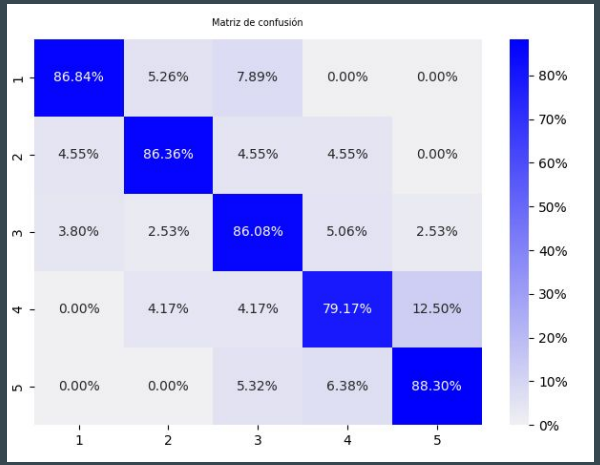
Matriz de confusión con distancias PESADAS

KNN con distancias pesadas 90/10

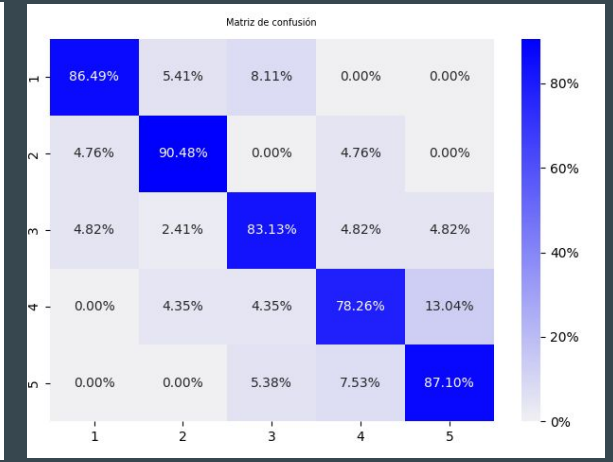
k = 5



k = 8



k=10



Precisión con distancias PESADAS

KNN con distancias pesadas 90/10

	Media	Desvío std
k = 5	0.86	0.06
k = 8	0.87	0.04
k = 10	0.85	0.09

Casos Particulares

Estudio de Normalización
Estudio eliminando NaN

Estudio de normalización

- Partición 80/20
- 5 vecinos
- Precisión:

	Pesado	Sin pesar
Normalizado	0.86	0.85
Sin normalizar	0.76	0.75

Estudio eliminando NaN

- Partición 80/20
- 5 y 8 vecinos
- KNN pesado
- Variables normalizadas

	Remueve	No remueve
k = 5	0.84	0.86
k = 8	0.83	0.84

¡Muchas gracias!

Dey, Patrick
Lombardi, Matías
Vázquez, Ignacio