



Algoritmos de Clasificación Supervisada

Grupo 3:

Mila Langone

Paula Oseroff

Luciana Diaz Kralj

Paula A. Domingues

Contenidos de la Presentación

01

Decision Tree

Ejercicio 1.a y 1.b

02

Random Forest

Ejercicio 1.c, 1.d y 1.e

03

Algoritmo kNN

Ejercicio 2

04

Conclusiones

Puntos interesantes
para destacar

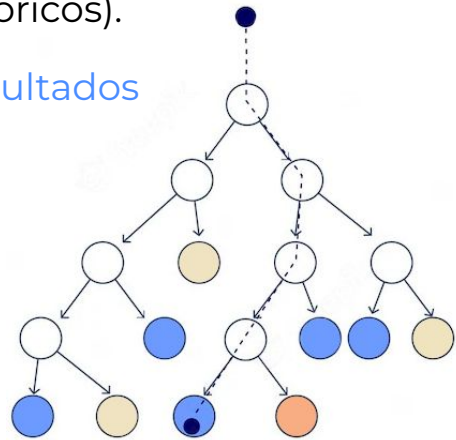


01

Decision Tree

¿Qué es un Árbol de Decisión?

- Es un algoritmo de **aprendizaje supervisado** no paramétrico que se utiliza para tareas de **clasificación**.
- Estructura jerárquica del proceso de aprendizaje representado en un árbol:
 - En sus **nodos intermedios** se realizan evaluaciones sobre los **atributos discriminantes** de la información (discretos o categóricos).
 - Los **nodos hoja** representan las **posibles clases o resultados** del conjunto de datos.



Cálculo de ganancia

- Se utiliza para seleccionar el **atributo más discriminante** A de la información.
- Elegimos la **entropía de Shannon** $H(X)$ para medir el **grado de desorganización** de los conjuntos de datos S_i , la cual devuelve un valor entre 0 y 1.

Información de ganancia:

$$Gain(S, A) = H(S) - \sum_{v \in \text{Valores}(A)} \frac{|S_v|}{S} H(S_v)$$

Siendo $H(X)$:

$$H(X) = - \sum_{i=1}^n P(x_i) \cdot \log_b P(x_i)$$

Algoritmo ID3

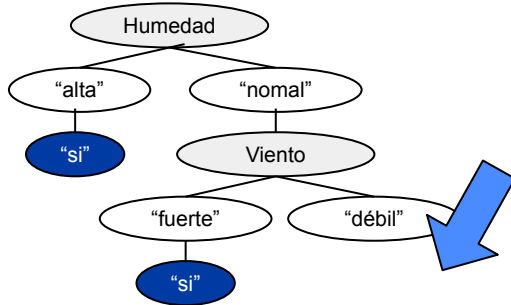
- El algoritmo **Iterative Dichotomiser 3** aprovecha la **entropía** y la **ganancia** de información como métricas de evaluación para el armado de árboles de decisión.
- Dos conjuntos de entrenamiento iguales resultan en un mismo árbol de decisión (es **determinístico**).
- Requiere una **discretización** o **categorización** previa de los atributos.

Algoritmo ID3

- Consta de los siguientes pasos:
 - Elegir el atributo que maximiza la **ganancia**
 - Crear tantas **ramificaciones** cómo posibles valores para ese atributo (**nodos valor**)
 - Separar el conjunto de datos según el **valor del atributo** para cada ramificación
 - Repetir los pasos anteriores hasta llegar a un **caso base**.
 - Agregar un nodo con la **clasificación** de ese subconjunto de datos del caso base llamado **nodo hoja**.

Algoritmo ID3

- Es **caso base** si:
 - Esa rama (raíz hasta nodo actual) contiene **todos los atributos**
 - Todos los datos en el conjunto tienen la **misma etiqueta de clasificación**
 - La separación del conjunto de datos de esa rama **no contiene datos**
 - Se cumple alguna condición de **pre-poda**



Humedad	Viento	¿Juega?
alta	débil	no
alta	fuerte	no

Humedad	Viento	¿Juega?
---------	--------	---------

Algoritmo ID3

- Condiciones de **pre-poda**:
 - Mínimo de **datos en el subconjunto** para hacer la separación,
 - Máximo número de **nodos hojas**,
 - Máxima **profundidad**,
 - Máximo **número de atributos** desde la raíz al nodo actual,
 - La ganancia del atributo no alcanza cierto **umbral**.

Ejercicio 1

Aplicar los algoritmos de [Decision Tree](#) y [Random Forest](#) para determinar si una persona devolverá el crédito o no.

- **Variable objetivo:**

'Creditability'

- **Variables explicativas:**

'Account Balance', 'Duration of Credit (month)', 'Payment Status of Previous Credit', 'Purpose', 'Credit Amount', 'Value Savings/Stocks', 'Length of current employment', 'Instalment per cent', 'Sex & Marital Status', 'Guarantors', 'Duration in Current address', 'Most valuable available asset', 'Age (years)', 'Concurrent Credits', 'Type of apartment', 'No of Credits at this Bank', 'Occupation', 'No of dependents', 'Telephone', 'Foreign Worker'

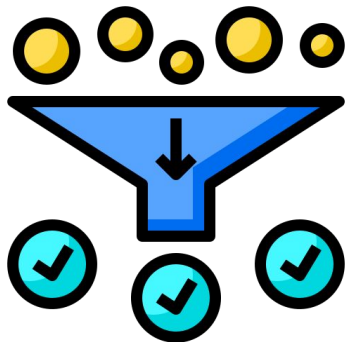


Pre-procesamiento del Dataset

1. Discretización

- Se asignan clases para agrupar los distintos rangos de valores que pueden tomar las variables.
N intervalos de igual amplitud.

5 cats: 'Credit Amount',
4 cats: 'Duration of Credit (month)',
3 cats: 'Age (years)'



2. División

- División del dataset en:
 - Train
 - Test
- Algoritmo K-Fold, $k = 3$ (667/333)

Análisis del Ejercicio: Clasificación

	Creditability	Creditability (predicted by DT)	A
81	1	1	1
510	1	1	1
628	1	1	1
828	0	1	1
569	1	1	1
30	1	1	1
38	1	1	1
648	1	1	0
13	1	1	0
818	0	1	1
...

Siendo:

1: **Devuelve**

0: **No devuelve**

Precisión (test):

Devuelve: 0.550

No devuelve: 0.615

Total (test):

Correctas: 214

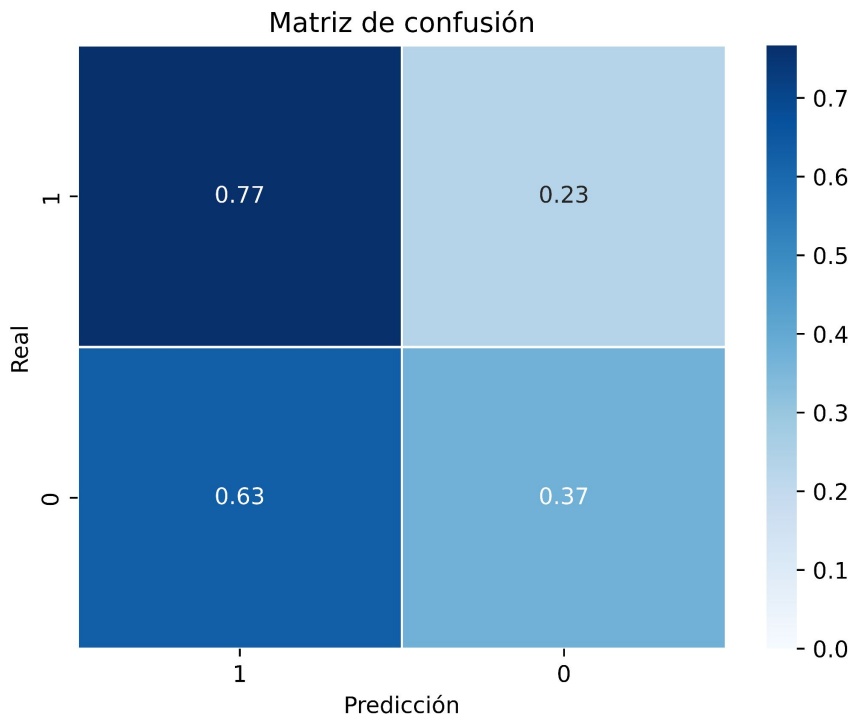
Incorrectas: 120

Conjunto elegido con validación

K-cruzada $K=3$

Análisis del Ejercicio:

Clasificación por Decision Tree



Siendo:

1: **Devuelve**

0: **No devuelve**

Precisión (test):

Devuelve: 0.550

No devuelve: 0.615

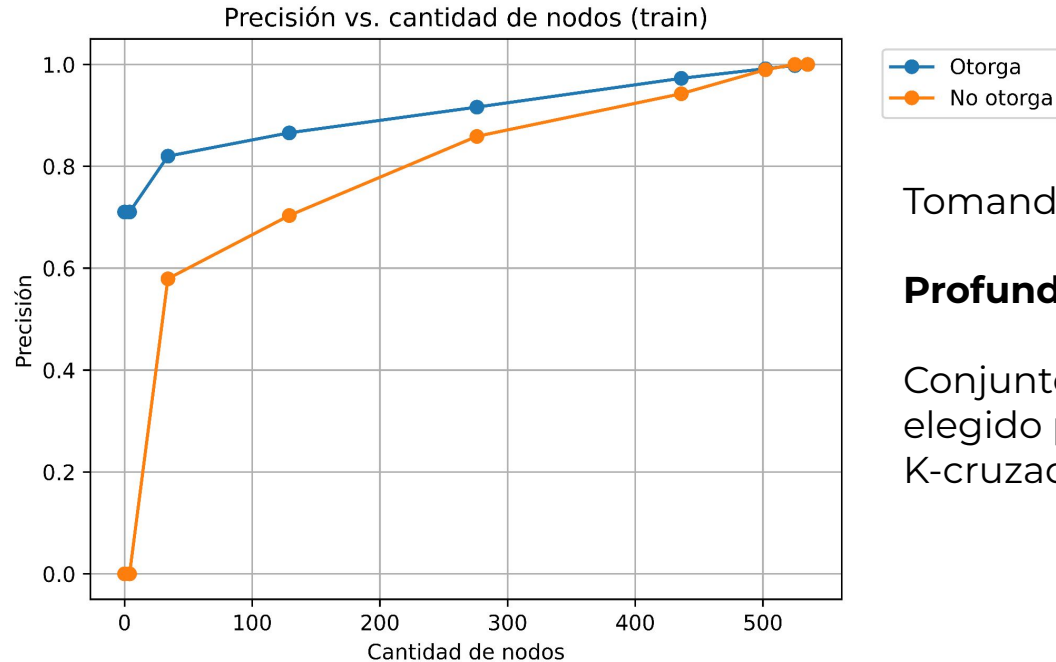
Total (test):

Correctas: 214

Incorrectas: 120

Conjunto elegido con validación
K-cruzada $K=3$

Análisis del Ejercicio: Cantidad de nodos

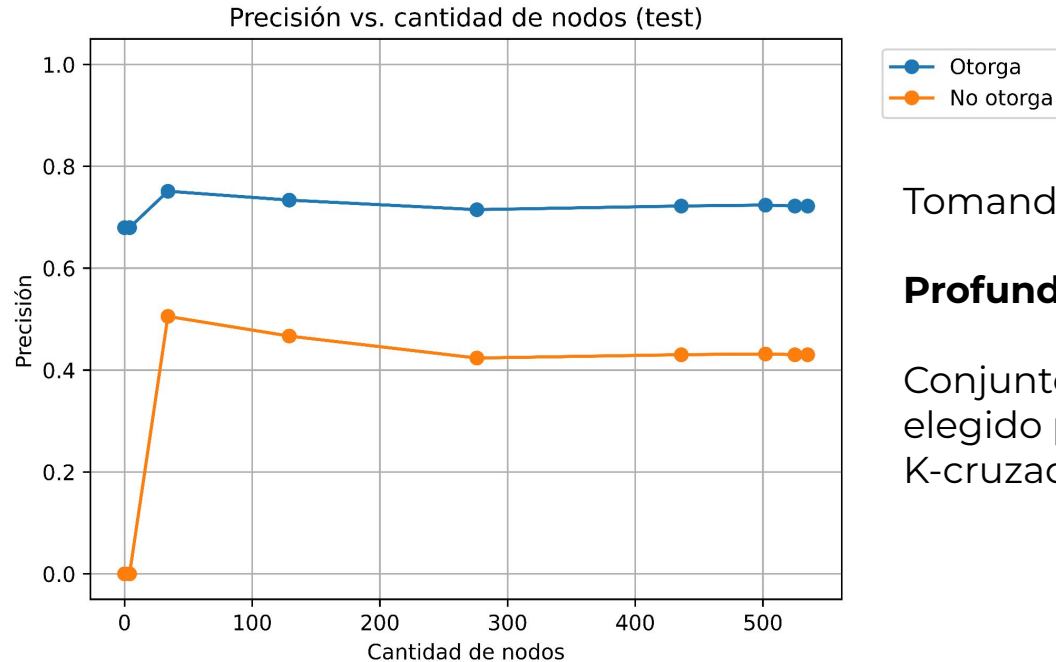


Tomando:

Profundidad = [0; 10]

Conjunto de datos
elegido por validación
K-cruzada con $K=3$

Análisis del Ejercicio: Cantidad de nodos



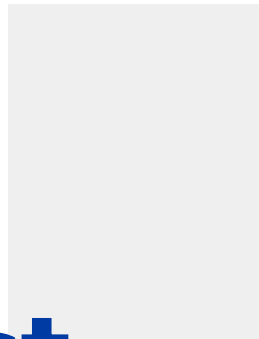
Tomando:

Profundidad = $[0; 10]$

Conjunto de datos
elegido por validación
K-cruzada con $K=3$

02

Random Forest



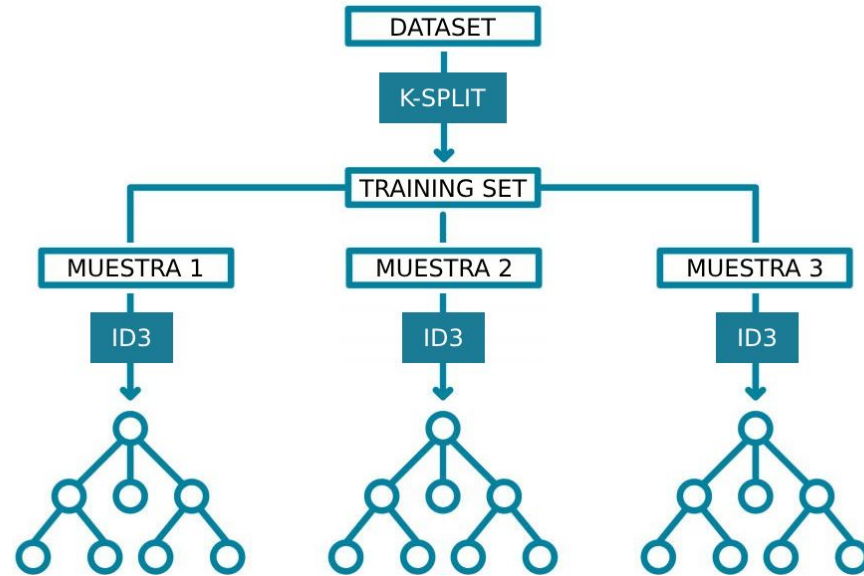
¿Qué es Random Forest?

- Los **árboles de decisión** tienen **limitaciones**:
 - Pueden generar **sobreajuste**, no generalizando bien los datos.
 - Pueden ser **inestables**, por lo que pequeñas variaciones de los datos de entrenamiento pueden resultar en árboles muy distintos.

Para solucionar estos problemas, se utiliza **Random Forest**

- Por medio de **bootstrapping**, se divide el conjunto de entrenamiento en muestras aleatorias con reposición.
- Se aplica **ID3** sobre cada subconjunto y se obtienen árboles de decisión ajustados a los datos que lo conforman.
- Las **predicciones** se obtienen **promediando** o sacando la moda de los resultados de cada árbol de decisión.

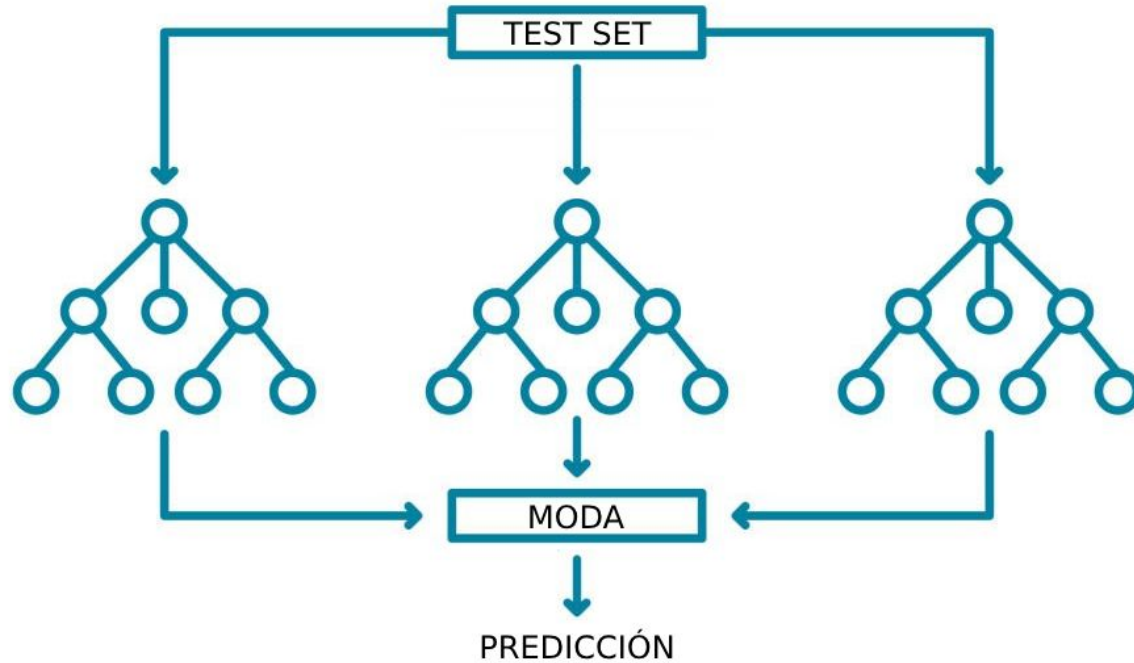
Armado de Random Forest



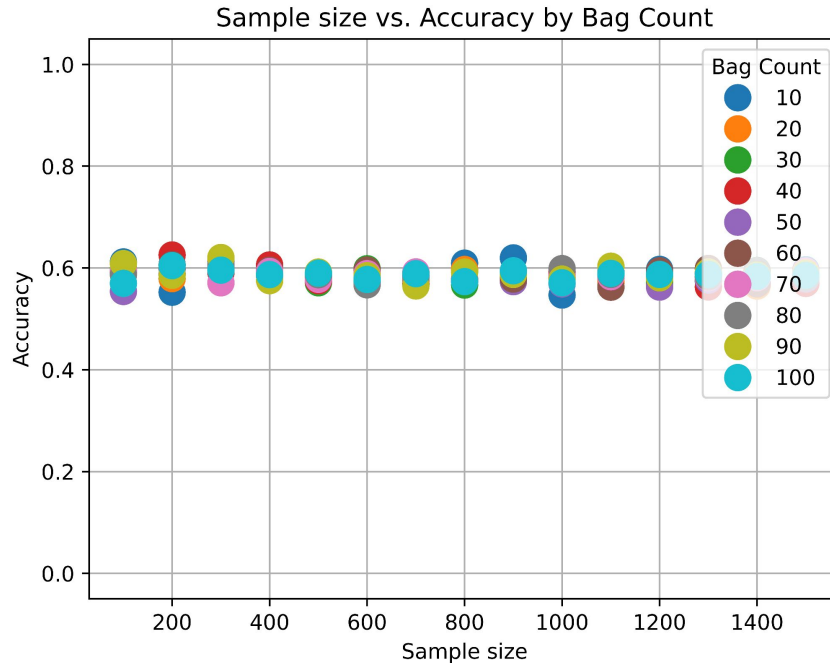
Parámetros

Cantidad de Bags
Tamaño de Muestras
Máxima Profundidad

Predicción de Random Forest



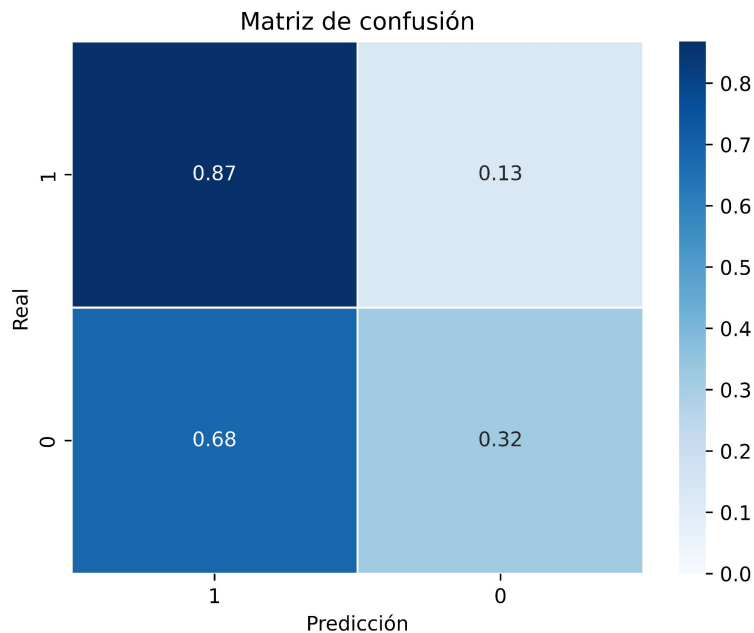
Análisis del Ejercicio: Random Forest Benchmark



Conjunto de datos elegido por validación K-cruzada con $K=3$

Análisis del Ejercicio:

Clasificación por Random Forest



Siendo:

1: **Devuelve**

0: **No devuelve**

Precisión (test):

Devuelve: 0.56

No devuelve: 0.71

Total (test):

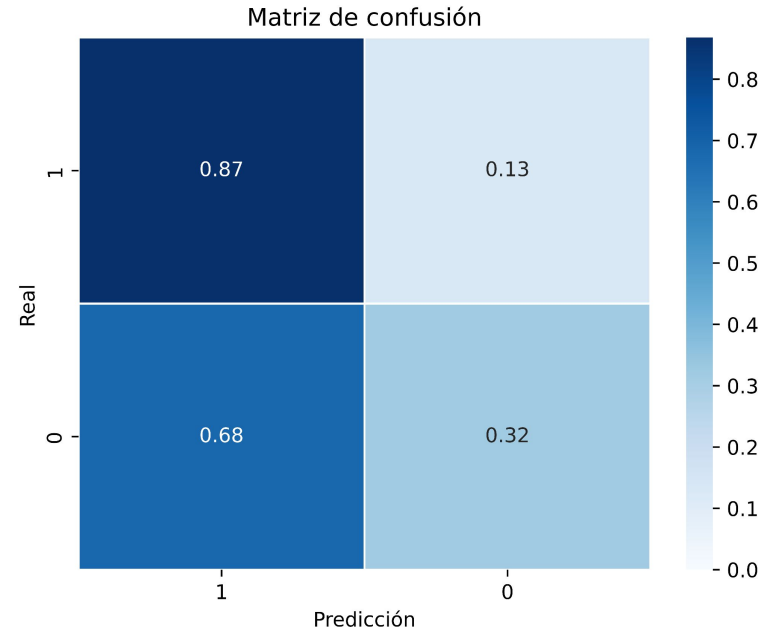
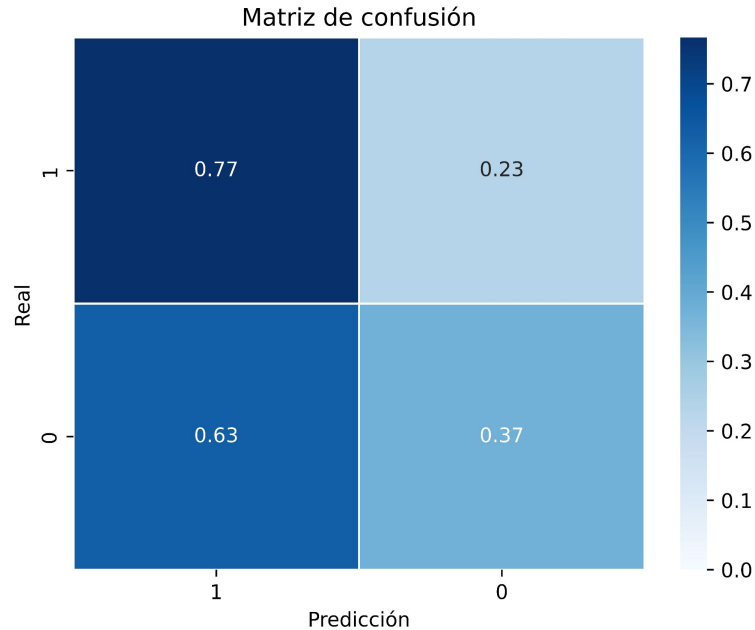
Correctas: 231

Incorrectas: 103

Parámetros: Bags: 40 Muestras: 200

Análisis del Ejercicio:

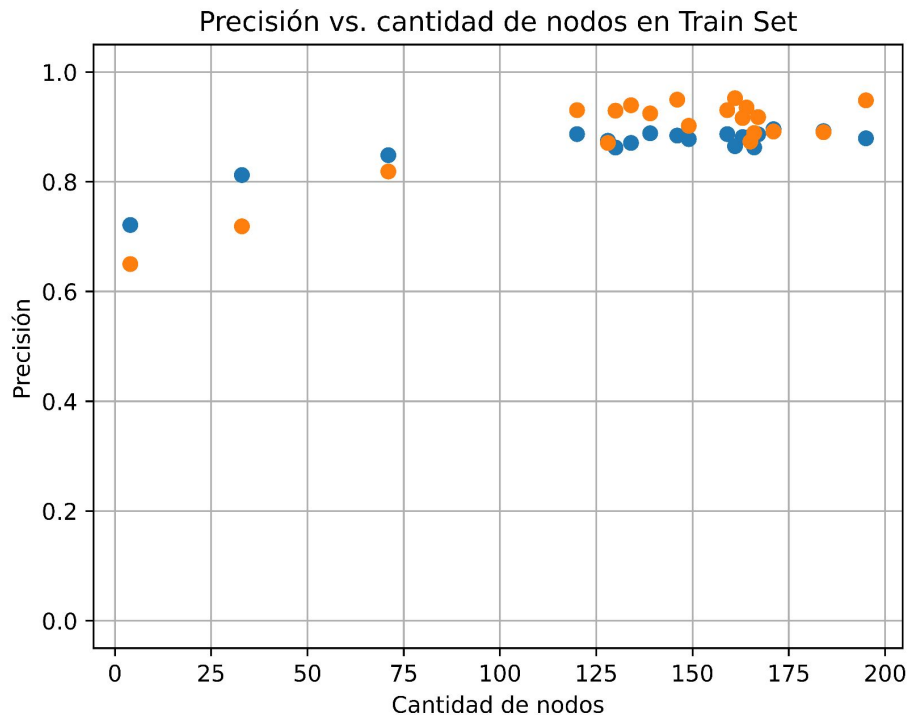
Árbol vs Bosque



Parámetros: Bags: 40 **Muestras:** 200

Análisis del Ejercicio:

Cantidad de nodos Train



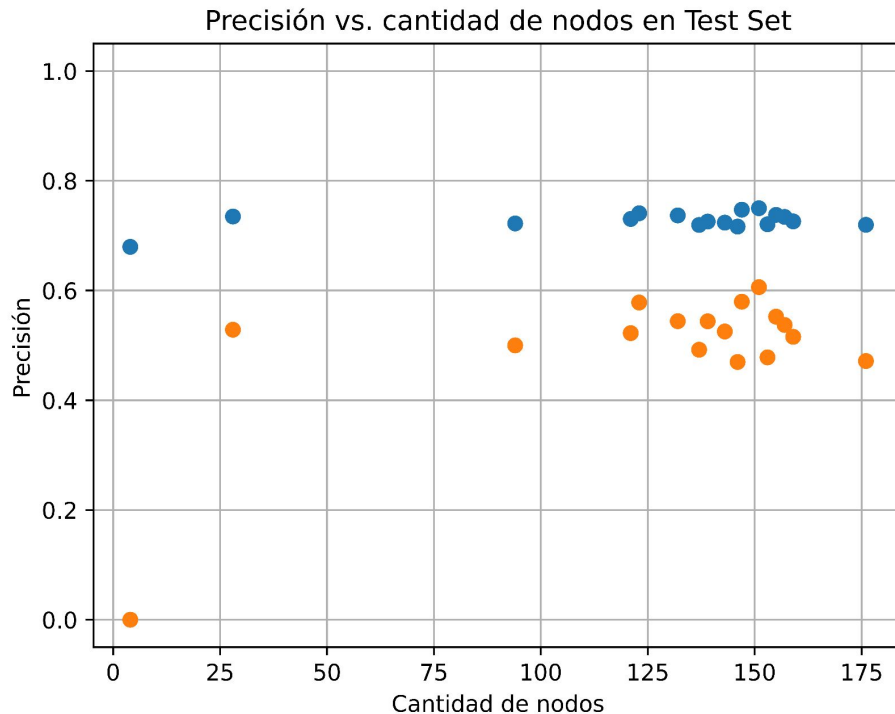
Bags: 40

Muestras: 200

Profundidad = [1; 20]

Conjunto de datos
elegido por validación
K-cruzada con $K=3$

Análisis del Ejercicio: Cantidad de nodos Test



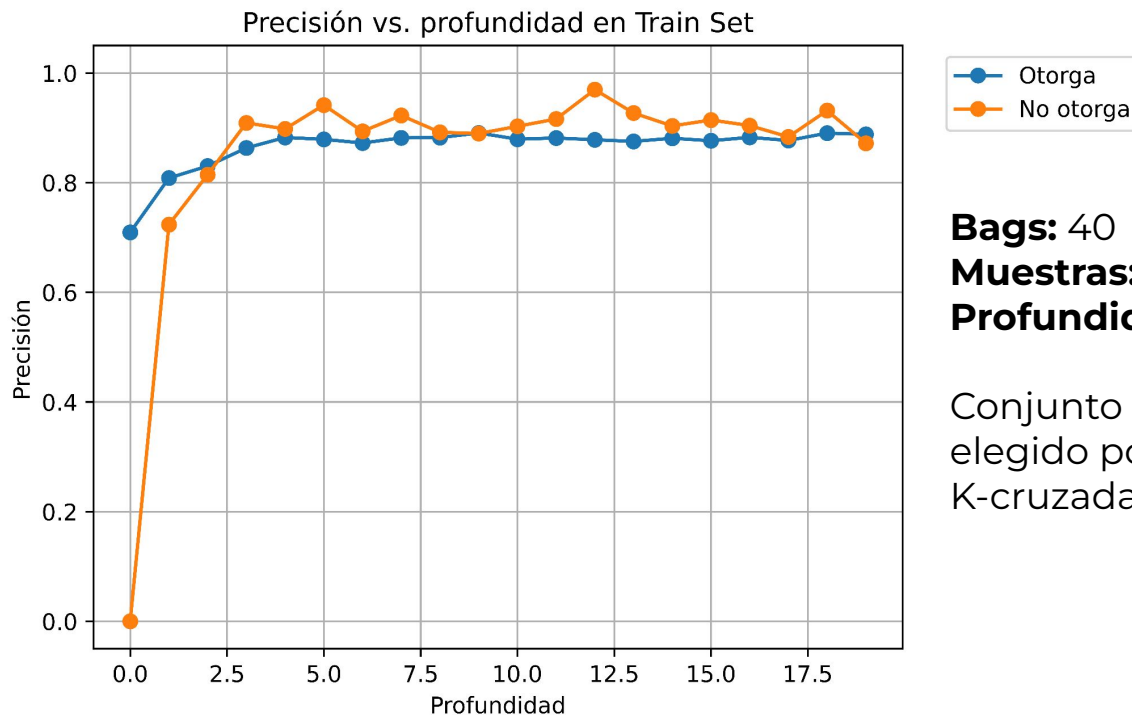
Bags: 40

Muestras: 200

Profundidad = [1; 20]

Conjunto de datos
elegido por validación
K-cruzada con $K=3$

Análisis del Ejercicio: Profundidad Train



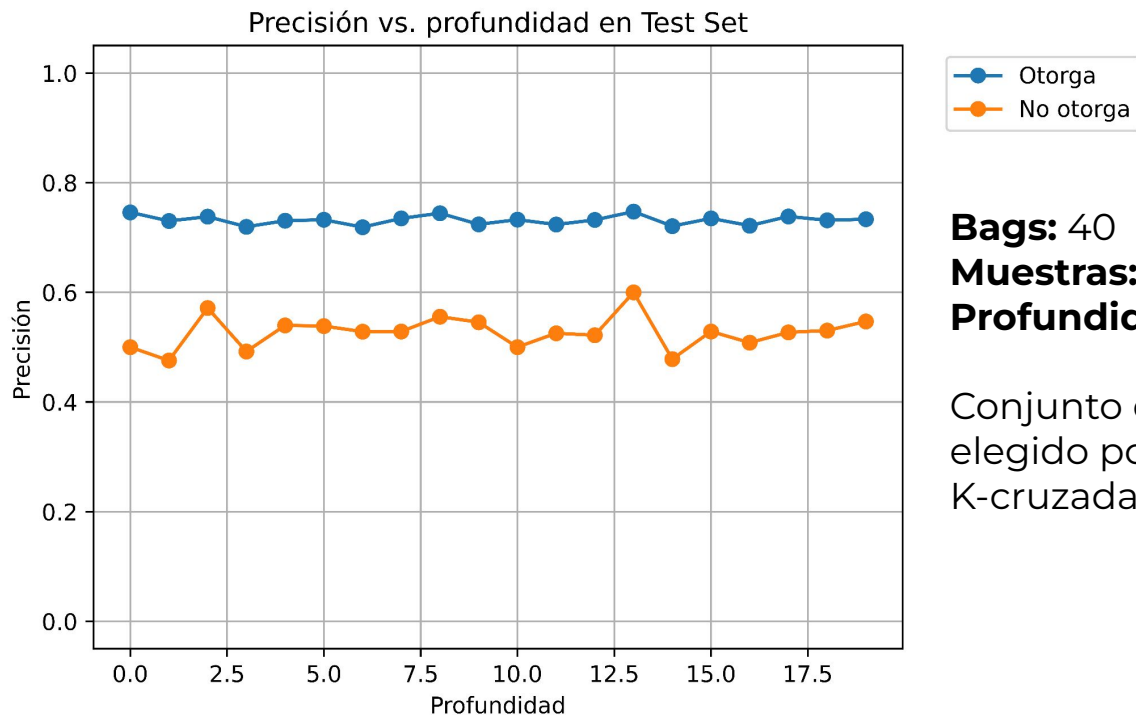
Bags: 40

Muestras: 200

Profundidad = [1; 20]

Conjunto de datos
elegido por validación
K-cruzada con $K=3$

Análisis del Ejercicio: Profundidad Test



Bags: 40

Muestras: 200

Profundidad = [1; 20]

Conjunto de datos
elegido por validación
K-cruzada con $K=3$



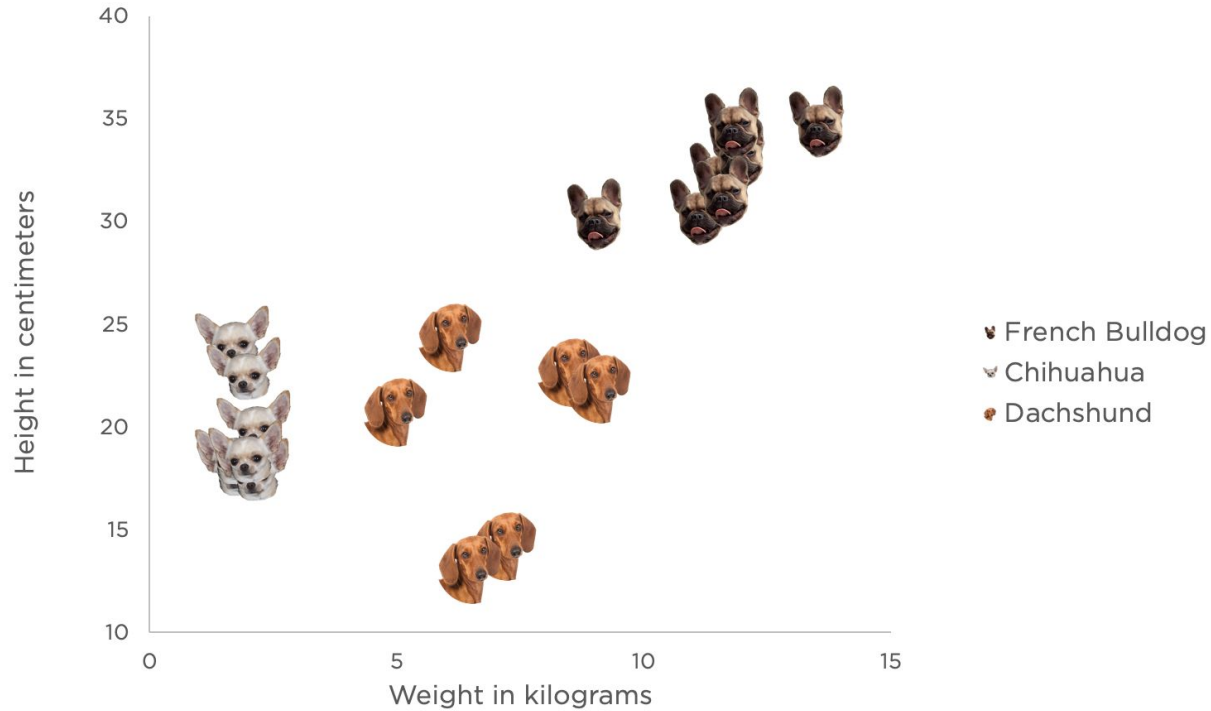
03

Algoritmo kNN

¿Qué es kNN?

- kNN es el acrónimo de "**k-Nearest Neighbors**".
- Es un algoritmo de **aprendizaje supervisado** utilizado para **clasificar** o **predecir** valores numéricos.
- Se basa en el principio de que los puntos de datos similares tienden a estar cerca en un espacio de características.
- Funciona encontrando los **k puntos** más cercanos a un nuevo dato y tomando decisiones basadas en la mayoría de etiquetas o promedio de valores entre esos vecinos.
- k es un parámetro crucial que determina la influencia de los vecinos en la clasificación.
 - **k pequeño**: modelo más sensible al ruido y a datos atípicos,
 - **k grande**: suaviza la frontera de decisión, modelo más robusto pero menos preciso.

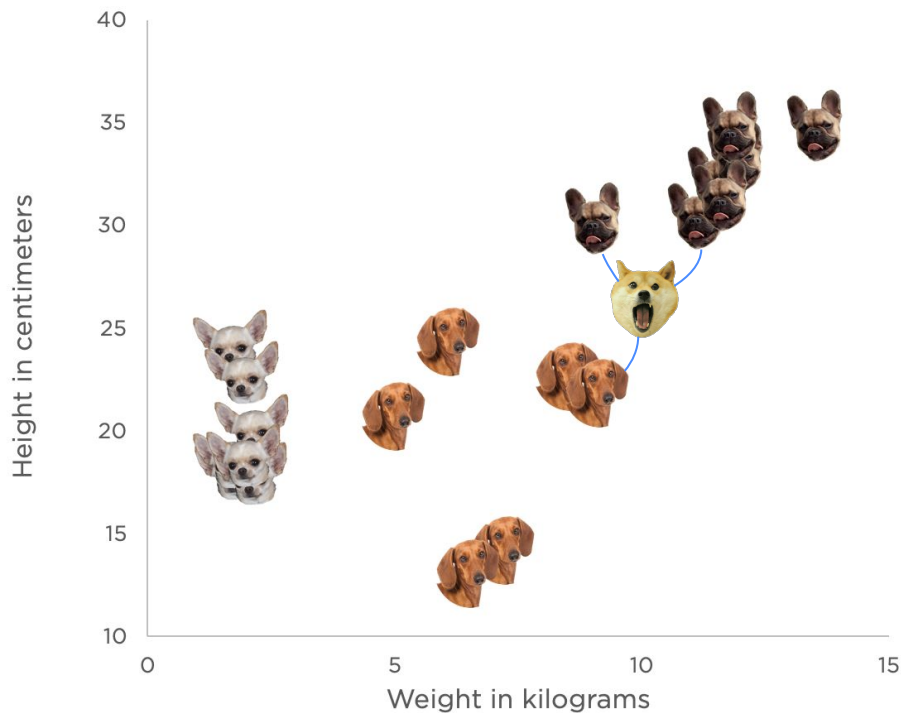
Intuición detrás del Algoritmo



Intuición detrás del Algoritmo



Intuición detrás del Algoritmo

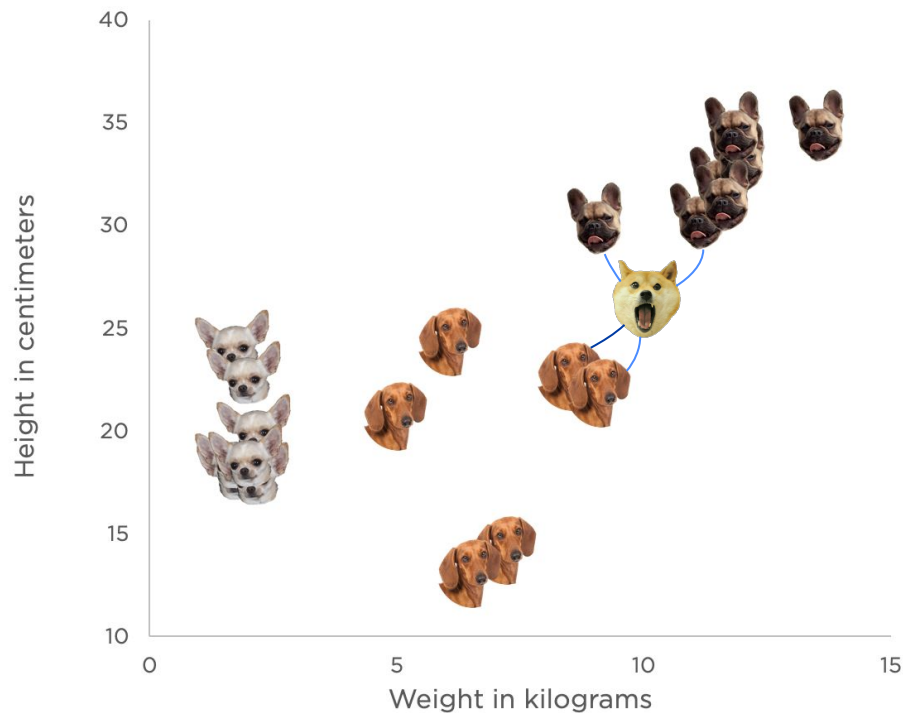


- 🐶 French Bulldog
- 🐶 Chihuahua
- 🐶 Dachshund
- 🐶 Shiba

Si $k = 3$, ¿Shiba = Frenchie?

¿Qué pasa para otros valores de k ?

Intuición detrás del Algoritmo

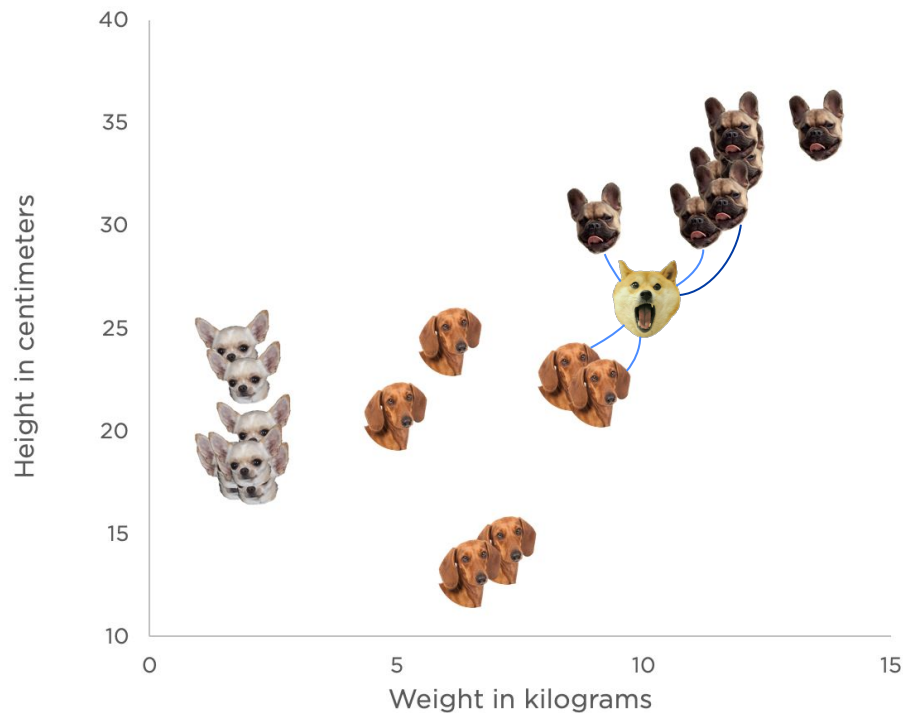


- 🐶 French Bulldog
- 🐶 Chihuahua
- 🐶 Dachshund
- 🐶 Shiba

¿Si $k = 4$?, empate.

Hay que incrementar k .

Intuición detrás del Algoritmo



- 🐶 French Bulldog
- 🐶 Chihuahua
- 🐶 Dachshund
- 🐶 Shiba

Finalmente, con $k = 5$,
vemos que otra vez,
Shiba = Bulldog Francés.

Pseudocódigo kNN

1. Se reciben los conjuntos de **train** y **test**, y el valor de **k**,
2. Por cada fila en test
 - a. Se calcula la distancia a los vecinos en train (**distancia euclídea**),
 - b. Se toman las k instancias a menor distancia,
 - c. Se cuentan las clases a las que pertenecen las instancias,
 - d. Se clasifica la nueva instancia en función a la clase con más apariciones.
En caso de empate, se regresa al ítem a y se aumenta k en uno.
 - e. Se adjunta la predicción a un array auxiliar
3. Se retorna el array auxiliar con todas las predicciones

Distancia Euclídea: define la distancia entre dos instancias. (a_r : valor del r-ésimo atributo de la instancia)

$$d(x_i, x_j) = \sqrt{\sum_{i=1}^n (a_r(x_i) - a_r(x_j))^2}$$

Weighted kNN

Los pasos son los mismos que el kNN simple, pero se **ponderan las distancias** en vez de contar únicamente la cantidad de apariciones:

$$w_i = \frac{1}{d(x_i, x_j)^2}$$

Y luego, al momento de contar las clases a las que pertenece cada instancia, se computa de la siguiente manera:

$$\hat{q}(x_q) = \arg \max_{v \in V} \sum_i w_i * 1_{\{v=f(x_i)\}}$$

En caso que la distancia sea **cero**, se le asigna la del valor **más cercano**.

Ejercicio 2

Review Title	Review Text	wordcount	titleSentiment	textSentiment	Star Rating	sentimentValue
Sin conexión	Hola desde (...)	23	negative	negative	1	-0,4864
faltan cosas	Han mejorado (...)	20	negative	negative	1	-0,5862
Version antigua	Me gustan (...)	17	NaN	negative	1	-0,6163
Esta bien	Sin (...)	6	negative	negative	1	-0,6518
Buena	Nada (...)	8	positive	negative	1	-0,7204
De gran ayuda	Lo (...)	23	positive	negative	1	-0,7268
Muy buena	Estaba (...)	16	positive	negative	1	-0,7368
...						
Hay que mejorar	No funcionan (...)	20	positive	negative	1	-0,1098
Muy bien ..pero	...	26	negative	negative	1	-0,1104

Aplicar los algoritmos **kNN** y **kNN con distancias pesadas** para clasificar las opiniones, utilizando distintos valores de k.


- **Variable objetivo:** 'Stars Rating'
- **Variables explicativas:** 'wordcount', 'titleSentiment', 'sentimentValue'.

Pre-procesamiento del Dataset

1. Limpieza

- Se eliminan las columnas repetidas,
- Si titleSentiment es NaN, entonces:
 - reemplazo "Star Rating" ≥ 3 ,
 - se droppea la fila.

3. División

- División del dataset en:
 - Train
 - Test
- k-folds, $k = 4$ (75/25) 

2. Normalización

- Normalización de "wordcount" y "sentimentValue"
- "titleSentiment" toma los valores:
 - 1 si "positivo",
 - 0 si "negativo".

Pre-procesamiento del Dataset

Tamaño del Dataset

Inicial: 257
Sin repetidos: 256
Sin NaN: 230

Average Word Count

➡ Star Rating 1: 12,2
Star Rating 2: 41,9
Star Rating 3: 8,08
Star Rating 4: 16,4
Star Rating 5: 4,29

Average Word Count*

➡ Star Rating 1: 12,4
Star Rating 2: 40,4
Star Rating 3: 8,37
Star Rating 4: 16,3
Star Rating 5: 4,3

Title Sentiment

Eliminando*
filas NaN { Positivos: 194
Negativos: 36

Positivos: 216
Negativos: 40

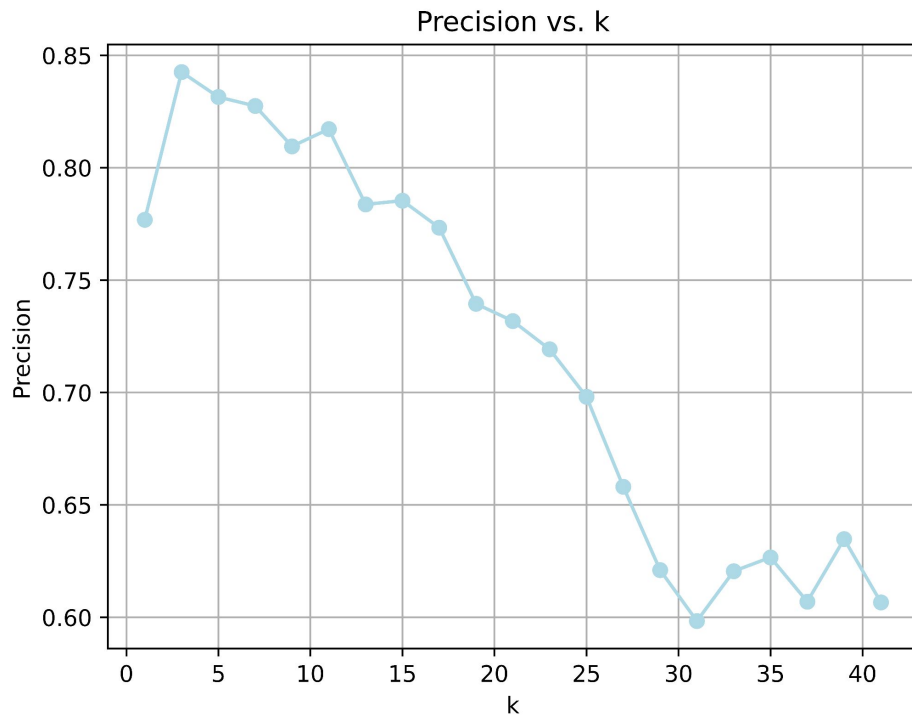
Star Rating Amount

Star Rating 1: 37
Star Rating 2: 24
Star Rating 3: 78
Star Rating 4: 30
Star Rating 5: 87

Star Rating Amount*

Star Rating 1: 34
Star Rating 2: 23
Star Rating 3: 69
Star Rating 4: 27
Star Rating 5: 77

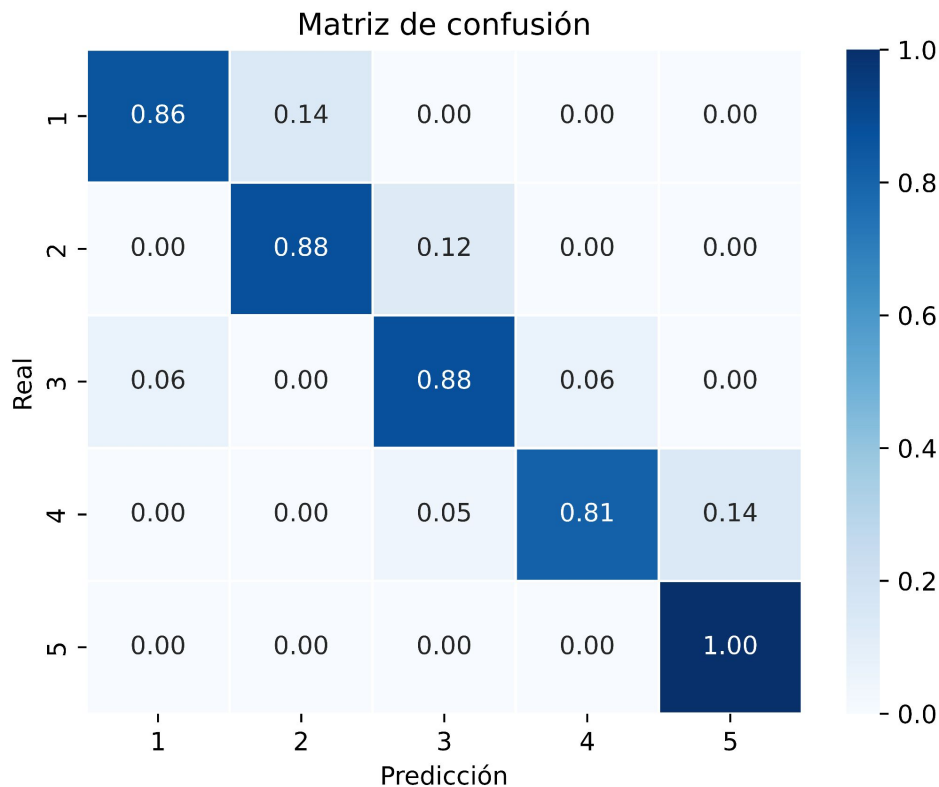
Análisis del Ejercicio: valores de k



Non-weighted kNN

- 10 k-folds con $k = 4$
- k óptimos = 3, 5, 7.

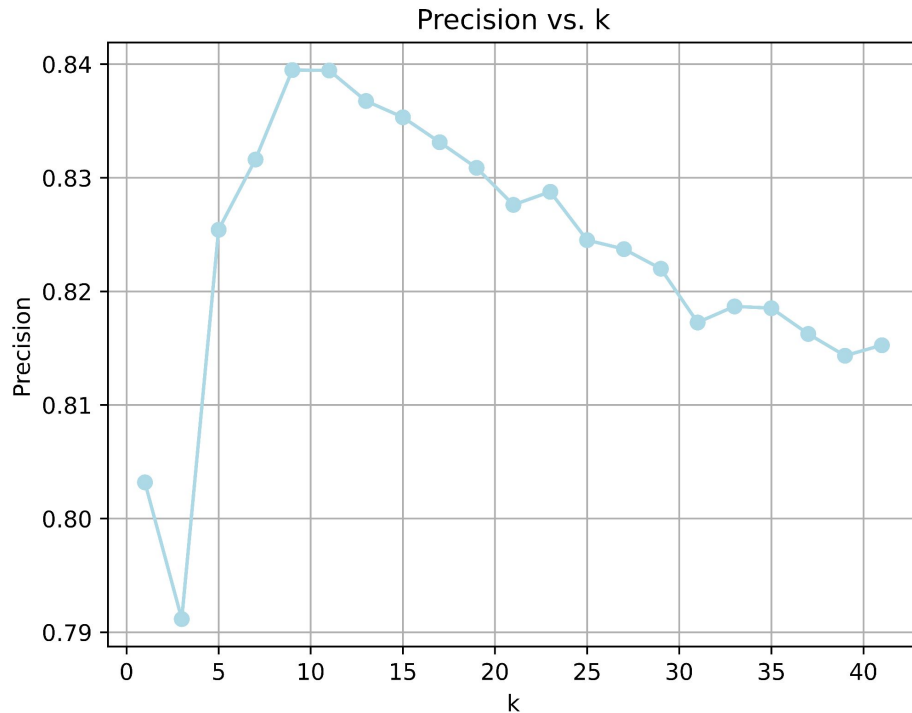
Análisis del Ejercicio: Matriz de Confusión



Non-weighted kNN

- Promedio de $k = [3, 5, 7]$.
- Precision: 0.86
- Accuracy: 0.94
- Std Precision: 0.07
- Std Accuracy: 0.02

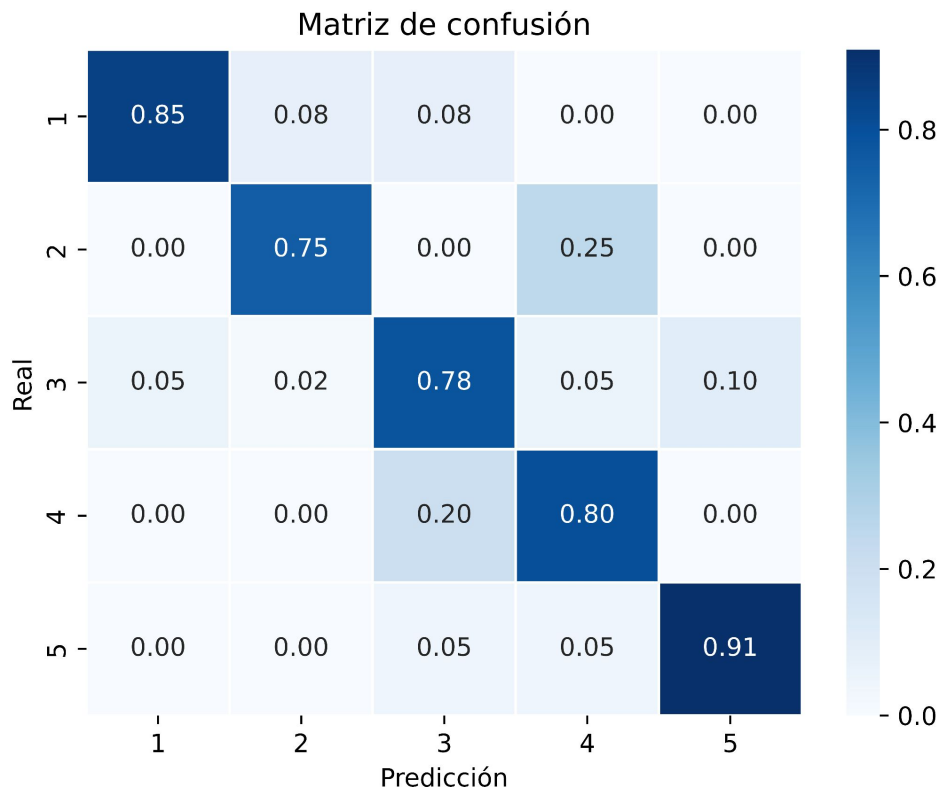
Análisis del Ejercicio: valores de k



Weighted kNN


- 10 k-folds con $k = 4$
- k óptimos = 9, 11, 13.

Análisis del Ejercicio: Matriz de Confusión



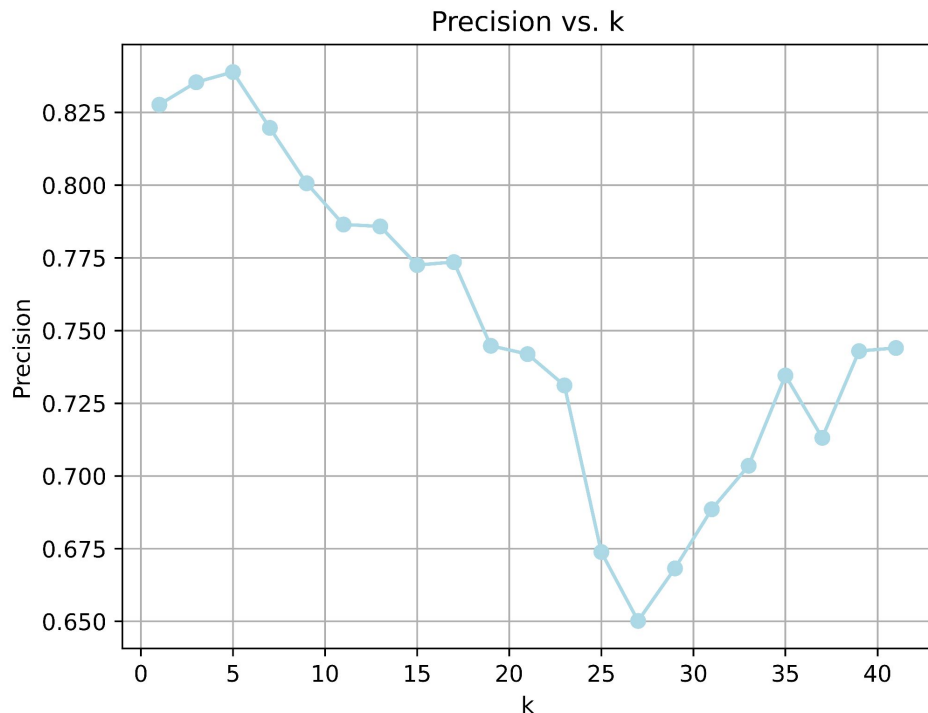
Weighted kNN

- Promedio de $k = [9, 11, 13]$.
- Precision: 0.89
- Accuracy: 0.95
- Std Precision: 0.04
- Std Accuracy: 0.01

A decorative graphic on the left side of the slide consisting of two blue squares. One is a medium blue square positioned higher and to the right, and the other is a darker blue square positioned lower and to the left, partially overlapping the first one.

¿Qué ocurre si en vez de reemplazar los valores NaN de “title Sentiment” los eliminamos del dataset?

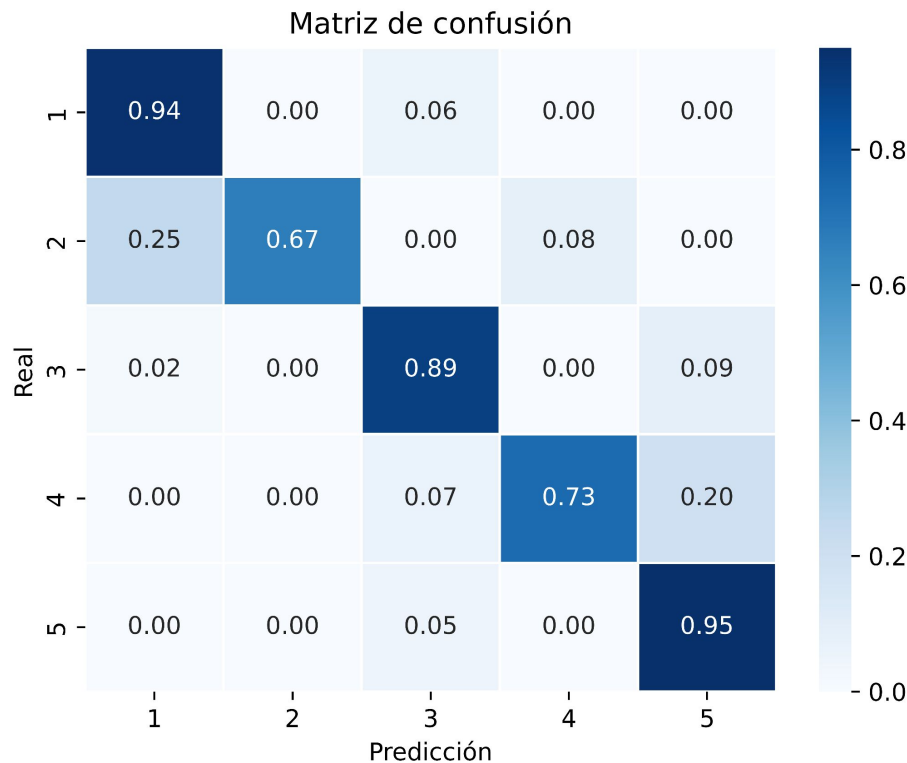
Análisis del Ejercicio: valores de k



Non-weighted kNN

- 10 k-folds con $k = 4$
- eliminando valores NaN
- k óptimos = 3, 5, 7.

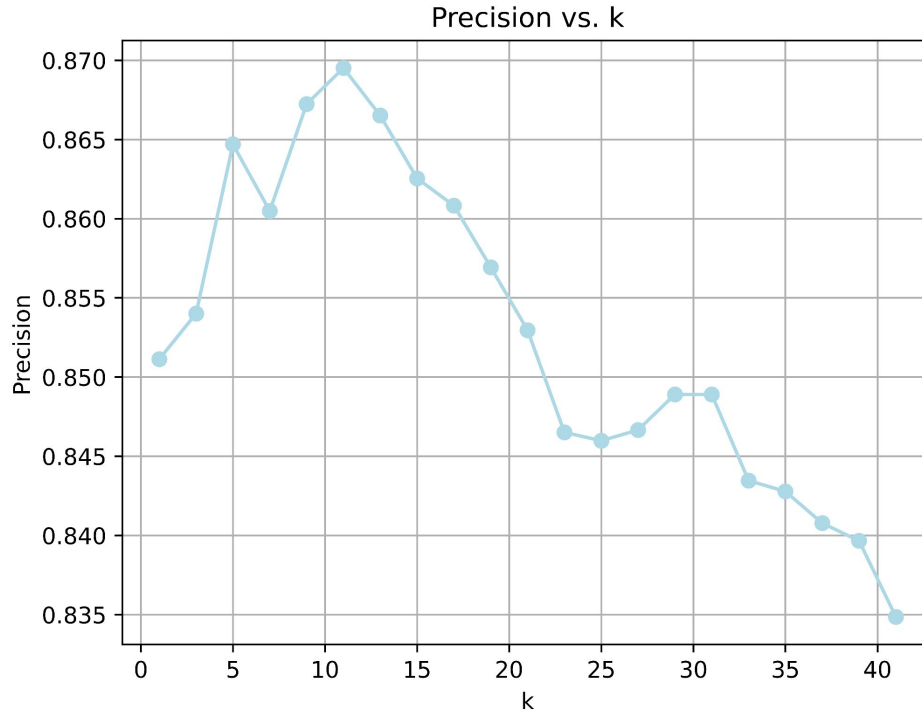
Análisis del Ejercicio: Matriz de Confusión



Non-weighted kNN

- Promedio de $k = [3, 5, 9]$.
- Precision: 0,85
- Accuracy: 0,93
- Std Precision: 0,08
- Std Accuracy: 0,005

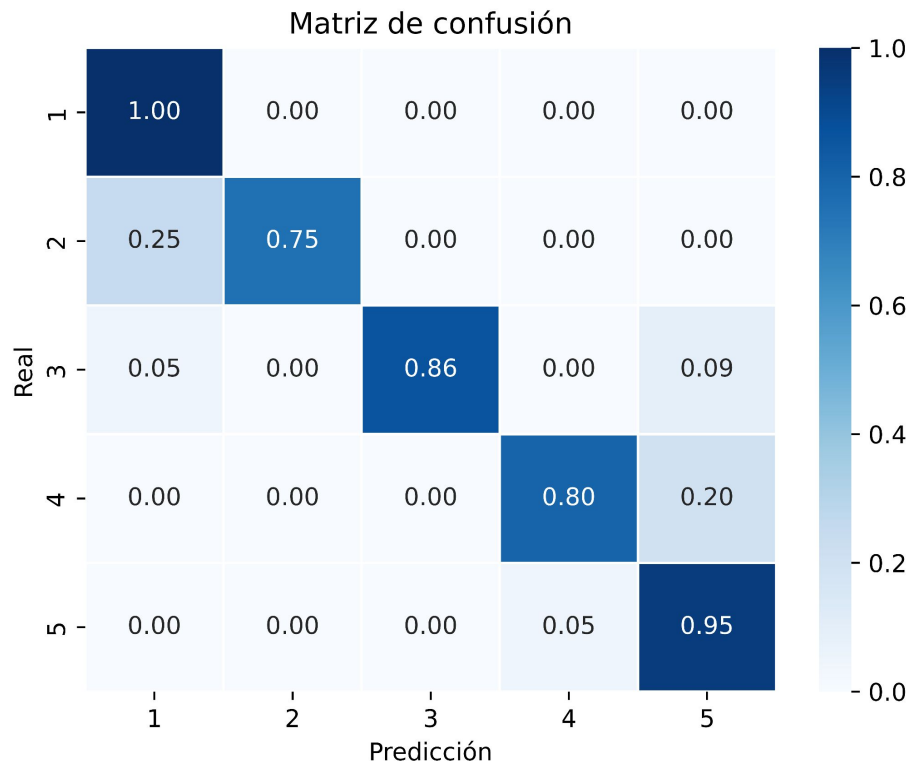
Análisis del Ejercicio: valores de k



Weighted kNN

- 10 k-folds con $k = 4$
- eliminando valores NaN
- k óptimos = 9, 11, 13.

Análisis del Ejercicio: Matriz de Confusión



Weighted kNN

- Promedio de $k = [9, 11, 13]$.
- Precision: 0,895
- Accuracy: 0,95
- Std Precision: 0,1
- Std Accuracy: 0,01

**¿Qué ocurre si no
normalizamos los valores
antes de ejecutar el
clasificador?**



Análisis del Ejercicio: sin Normalizar

Probamos la configuración inicial ($k_folds = 4$ y reemplazando los NaN) y comparamos la **precisión** del clasificador:

Non-weighted:

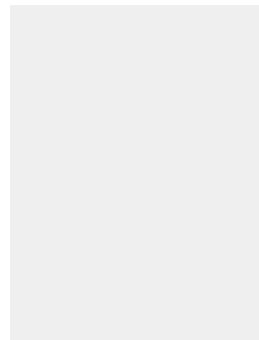
	Normalized	Non-Normalized
Precision	0.86	0.68
Std Deviation	0.07	0.21

Weighted:

	Normalized	Non-Normalized
Precision	0.89	0.74
Std Deviation	0.04	0.20

04

Conclusiones



Nuestras Conclusiones

- Una buena **categorización previa** de los datos es fundamental para obtener un buen árbol (o bosque). Que tan buenos resultados de clasificación se obtienen dependerá de las mismas.
- Un árbol de menor altura (y por ende, menor cantidad de nodos) puede tener una mejor precisión que uno de mayor altura (es **más general**).
- Elegir una relación correcta entre **cantidad de muestras** y **cantidad de árboles** tiene un alto impacto en la **accuracy** del modelo.
- Al depender de el conjunto que se utiliza para armarlos, que conjunto elegir y cómo seleccionarlo va a definir que tan bien se ajusta al problema.
- En la data hay un claro **sesgo hacia no otorgar créditos**.

Nuestras Conclusiones

- Una previa **normalización** de los datos es fundamental para el correcto funcionamiento del algoritmo kNN.
- **Weighted-kNN** tiene un **mejor desempeño** que non-weighted kNN sin considerar un valor de k en particular.
 - Weighted kNN se mantiene siempre en una precisión mayor al 80%, llegando casi al 90% para algunos valores.
 - Non-weighted kNN presenta picos de hasta 85% y mínimos del 60%.
- Si bien los valores máximos que toma k son similares, **non-weighted** presenta **mínimos de precisión** mucho más bajos.
- No reemplazar los valores de NaN genera una pequeña mejoría de los resultados → esto podría indicar que el criterio seleccionado para clasificar estos valores no fué lo suficientemente preciso.

¡Gracias!

¿Preguntas?

Grupo 3:

Mila Langone
Paula Oseroff
Luciana Diaz Kralj
Paula A. Domingues