

# K SCHOOL

Master en Data Science

Francisco José Sánchez Martín



# ¿Quién soy?: Paco Sánchez



Francisco José Sánchez Martín  
Data & Analytics Responsible  
The Heineken Company

## Agenda

### 1- Clasificación vs Regresión

---

### 2- Árboles de decisión

---

### 3- Bosques

---

### 4- K-vecinos

---

### 4- Ensembles

---

### 6- Fases de un proyecto

---

### 7- Ejercicios guiados

## Agenda

**1.1- Clasificación**

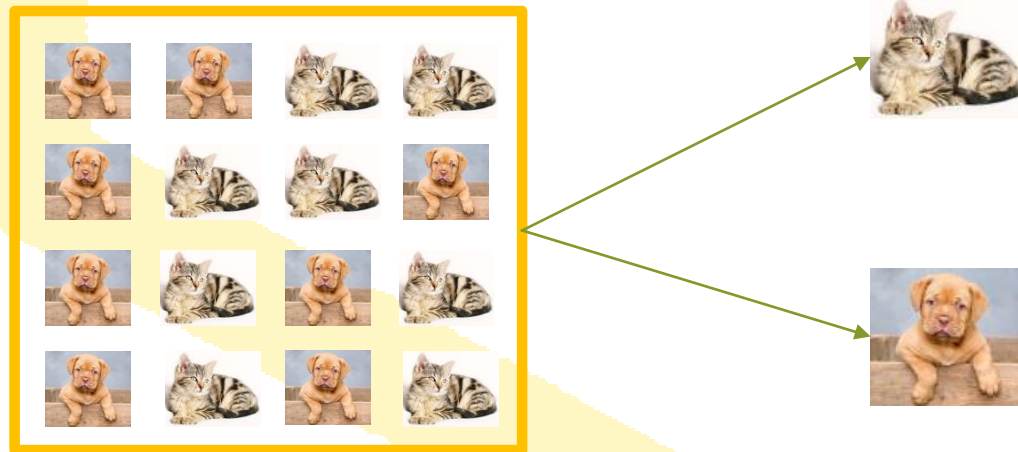
---

1.2- Regresión

---

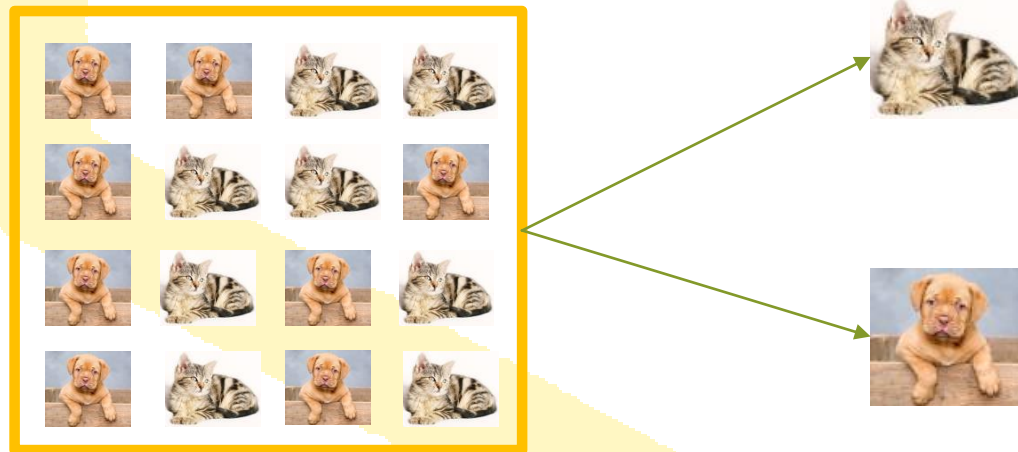
# Modelos de Clasificación

En los problemas de clasificación, el objetivo es encontrar una función que asigna a cada uno de los registros, obtenidos en el output, una clase o una etiqueta.



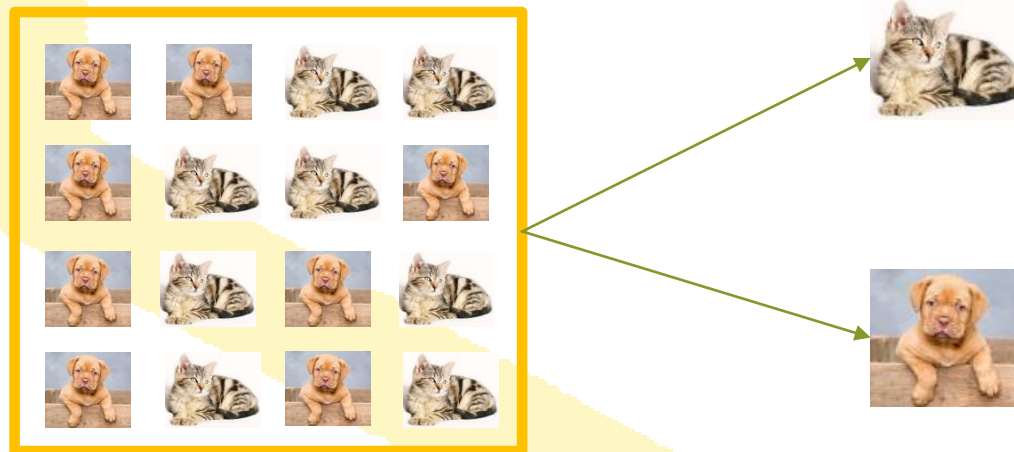
# Modelos de Clasificación

En modelos de aprendizaje automático supervisado de clasificación la variable objetivo a predecir es una determinada clase, es decir, una variable discreta.



# Modelos de Clasificación

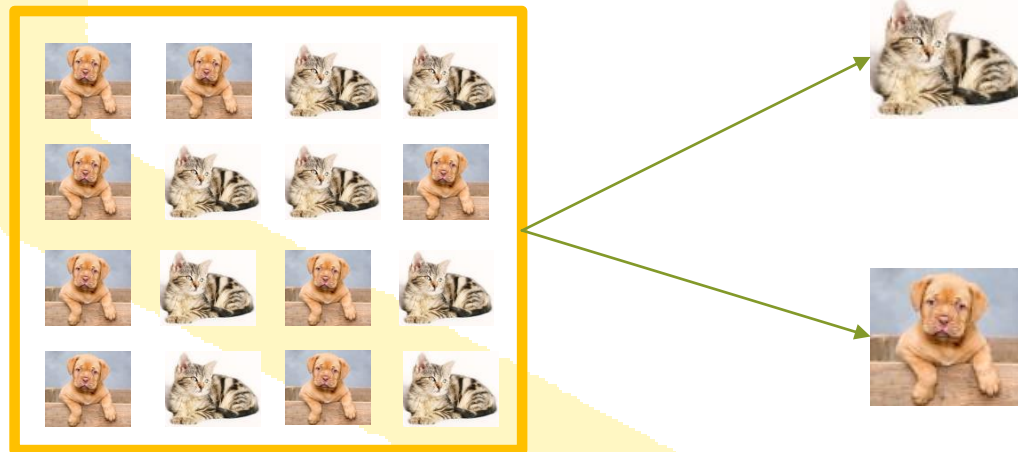
Un modelo de aprendizaje automático supervisado de clasificación, es un modelo basado en el cálculo de probabilidad, es decir, cual es la clase más probable para cada registro.



# Modelos de Clasificación

Ejemplos de variables objetivo para modelos de regresión:

Abandono (Si/No), Compra (Si/No), Spam (Si/No), Sexo (H/M), etc...





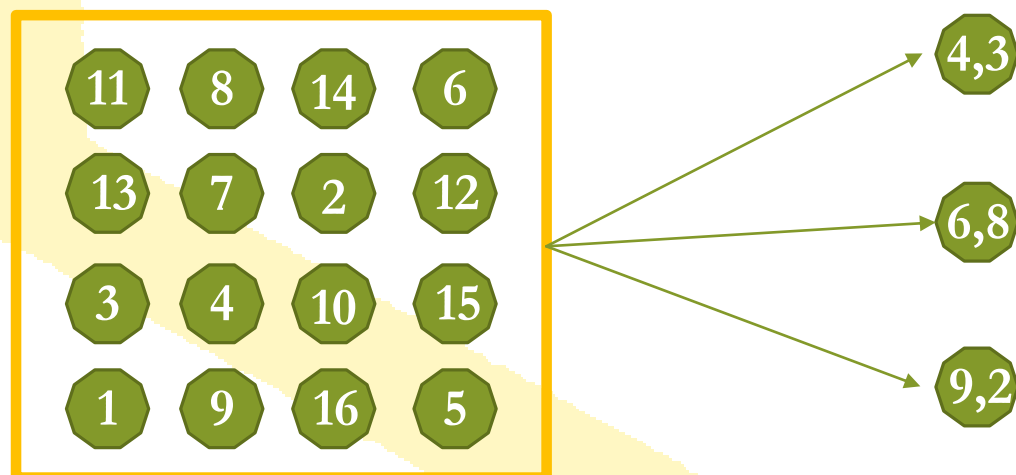
## Agenda

1.1- Clasificación

**1.2- Regresión**

# Modelos de Regresión

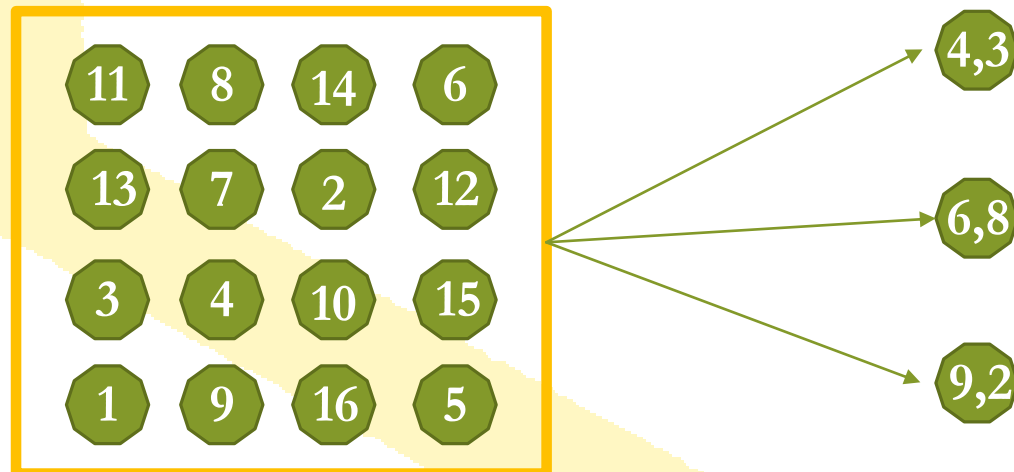
En los problemas de regresión, el objetivo es encontrar una función que asigne a cada uno de los registros, obtenidos en el output, un valor estimado sobre la variable objetivo.



# Modelos de Regresión

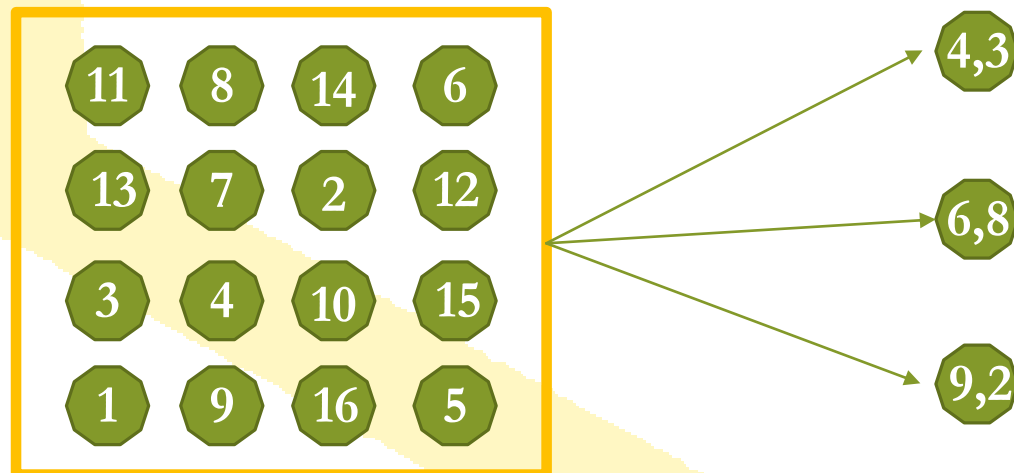
Esta categorización se aplica únicamente a los modelos basados en aprendizaje supervisado.

En modelos de aprendizaje automático supervisado de regresión la variable objetivo a predecir es numérica y continua.



# Modelos de Regresión

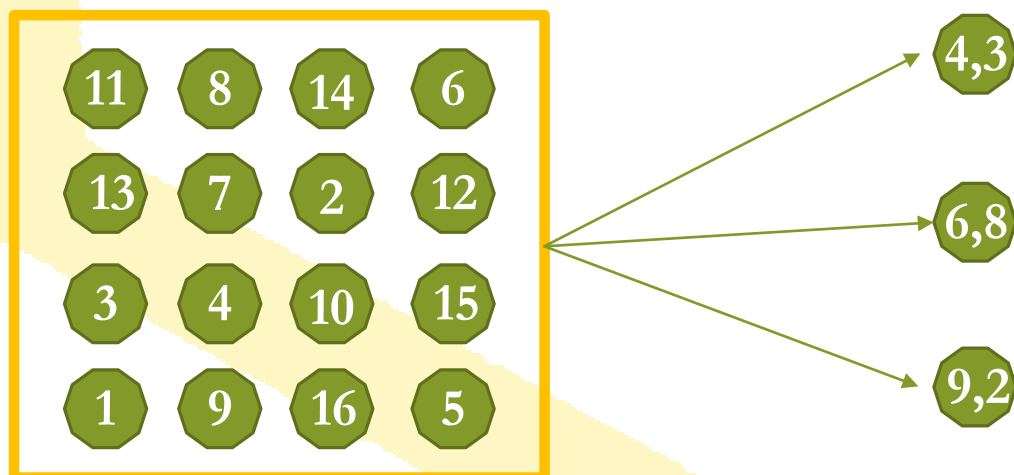
¡OJO! La variable objetivo **NO** tiene porque ser continua en el dataset de entrenamiento, pero en el output de las predicciones **SI** lo será!



# Modelos de Regresión

Ejemplos de variables objetivo para modelos de regresión:

Precio de vivienda, nota media de un alumno, números de alumnos matriculados, esperanza de vida, edad de una persona, etc...



## Agenda

1- Clasificación vs Regresión

---

**2- Árboles de decisión**

---

3- Bosques

---

4- K-vecinos

---

4- Ensembles

---

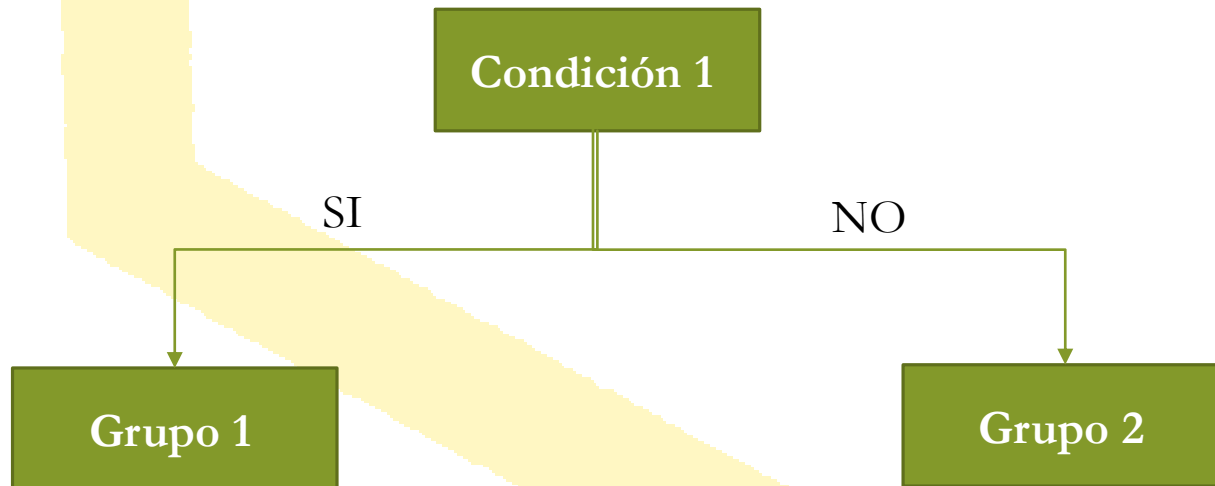
6- Fases de un proyecto

---

7- Ejercicios guiados

# Árboles de decisión

- Los árboles de decisión crean estructuras similares a los sistemas de decisión basados en reglas.



## Agenda

### 2.1- Árbol de Clasificación

---

### 2.2- Árboles de Regresión

---

### 2.3- Criterios de creación de un árbol

---

### 2.4- Métricas

---

### 2.5- Criterio detener un árbol

---

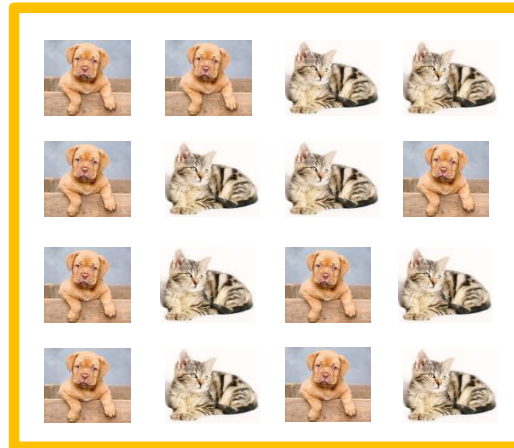
### 2.6- Criterio de decidir la predicción

---

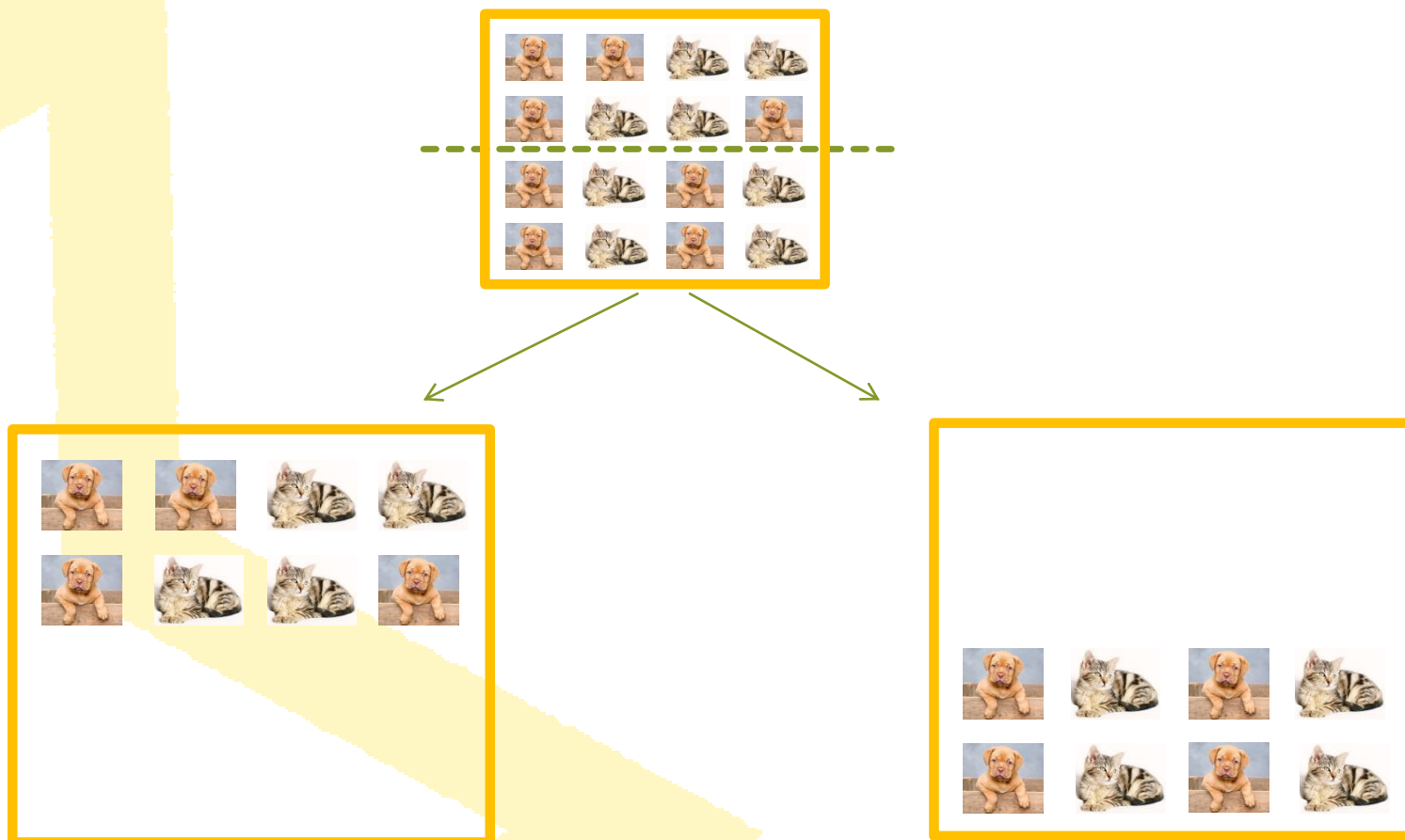
### 2.7- Cortes avanzados



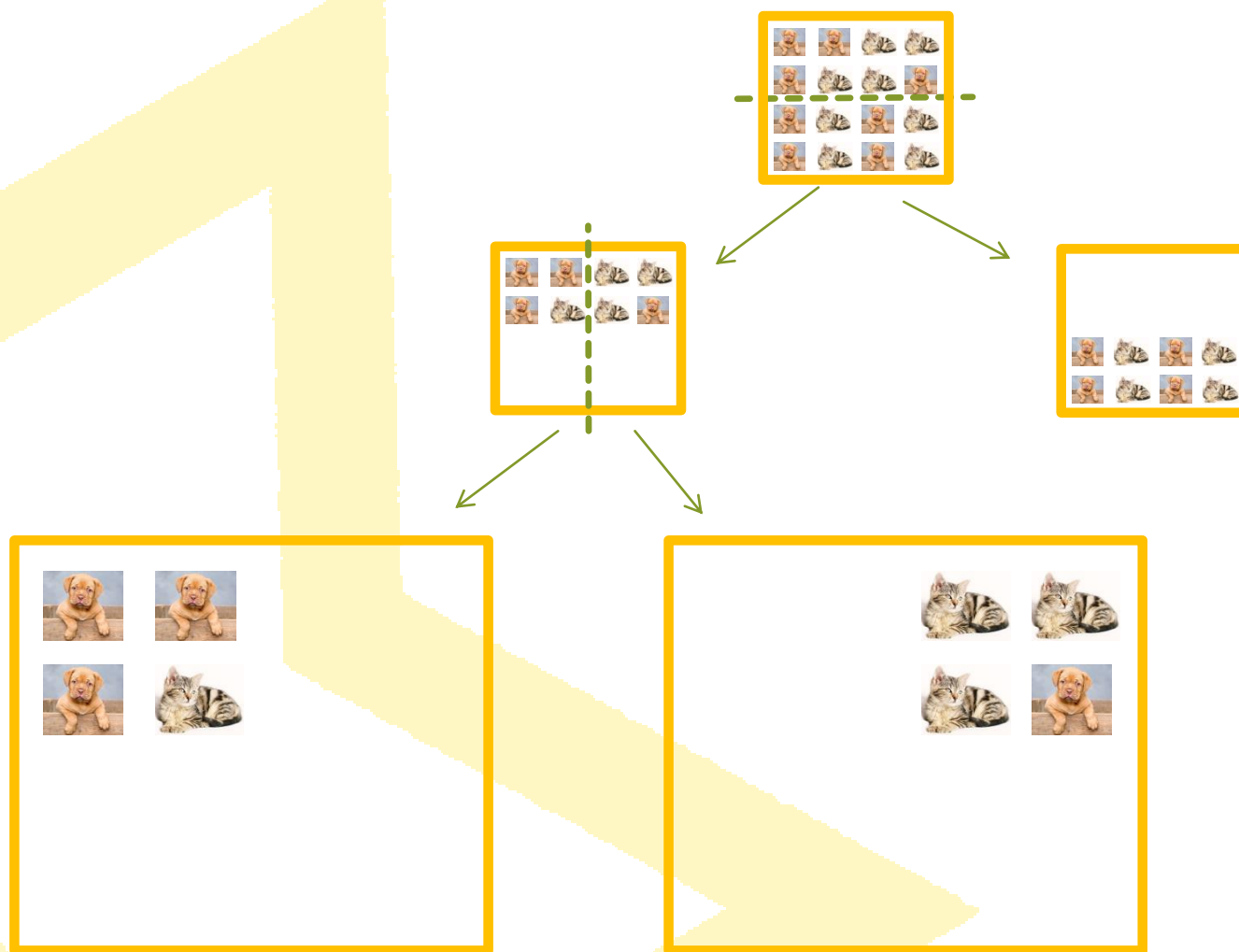
# Árbol de clasificación



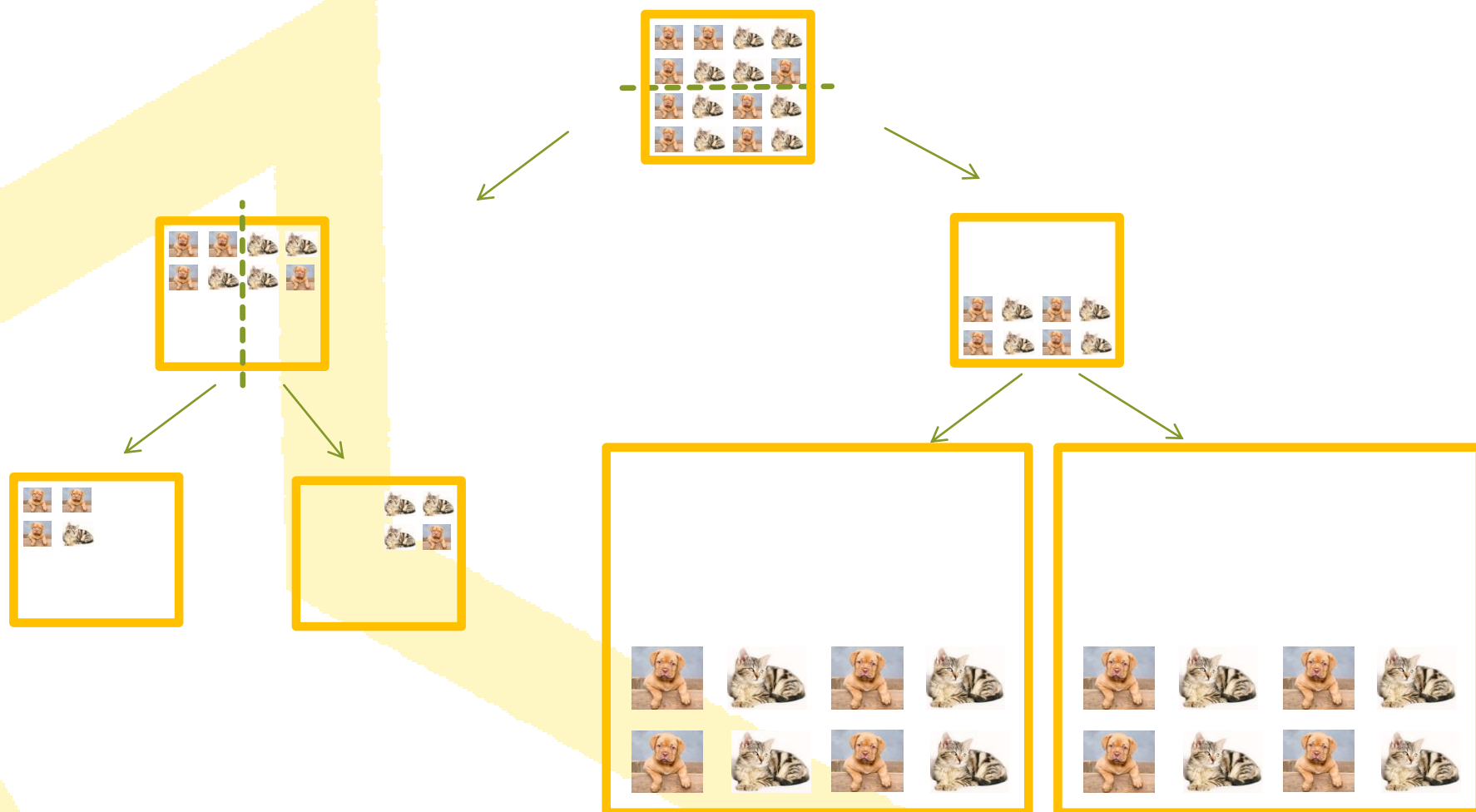
# Árbol de clasificación



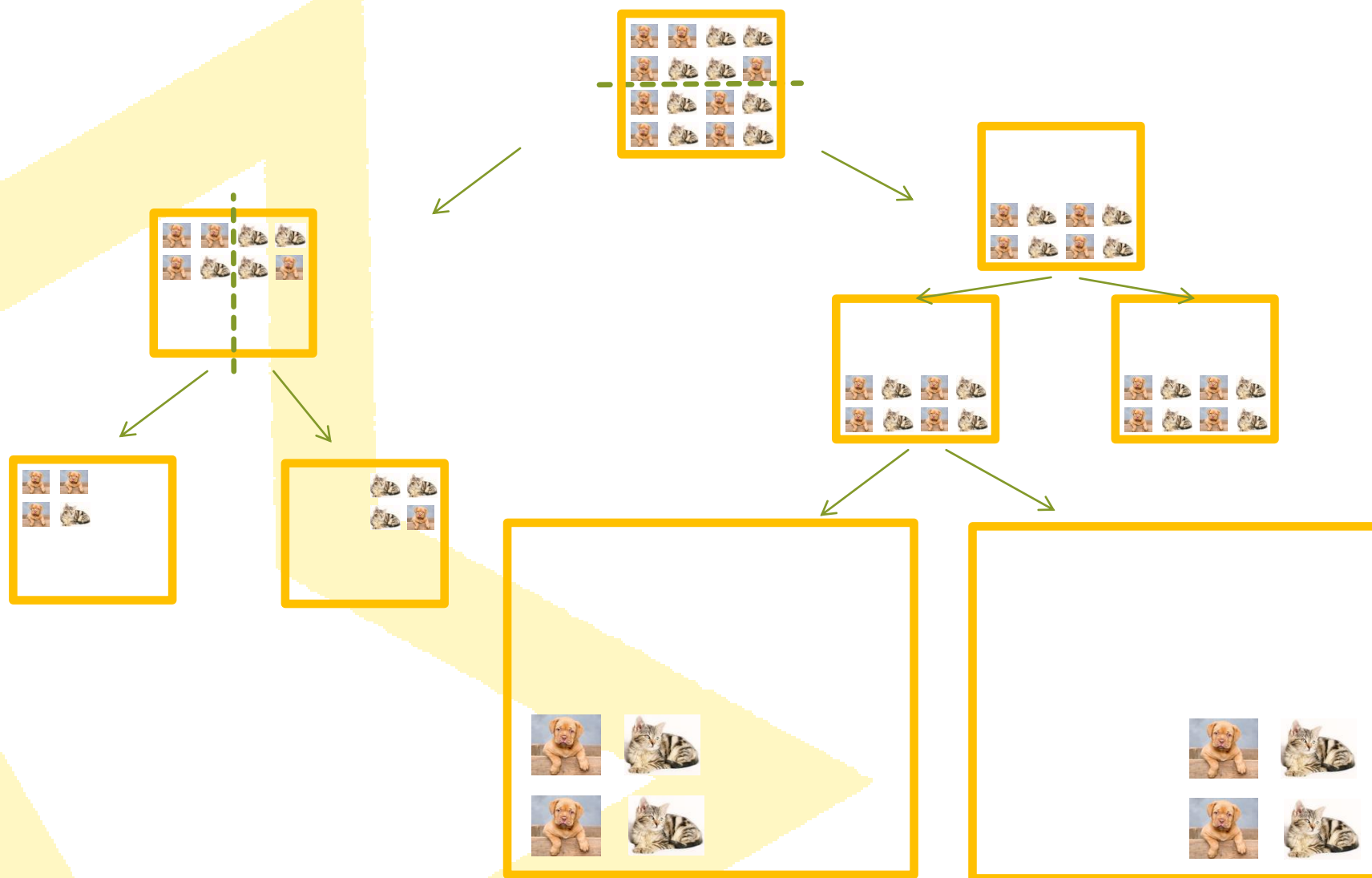
# Árbol de clasificación



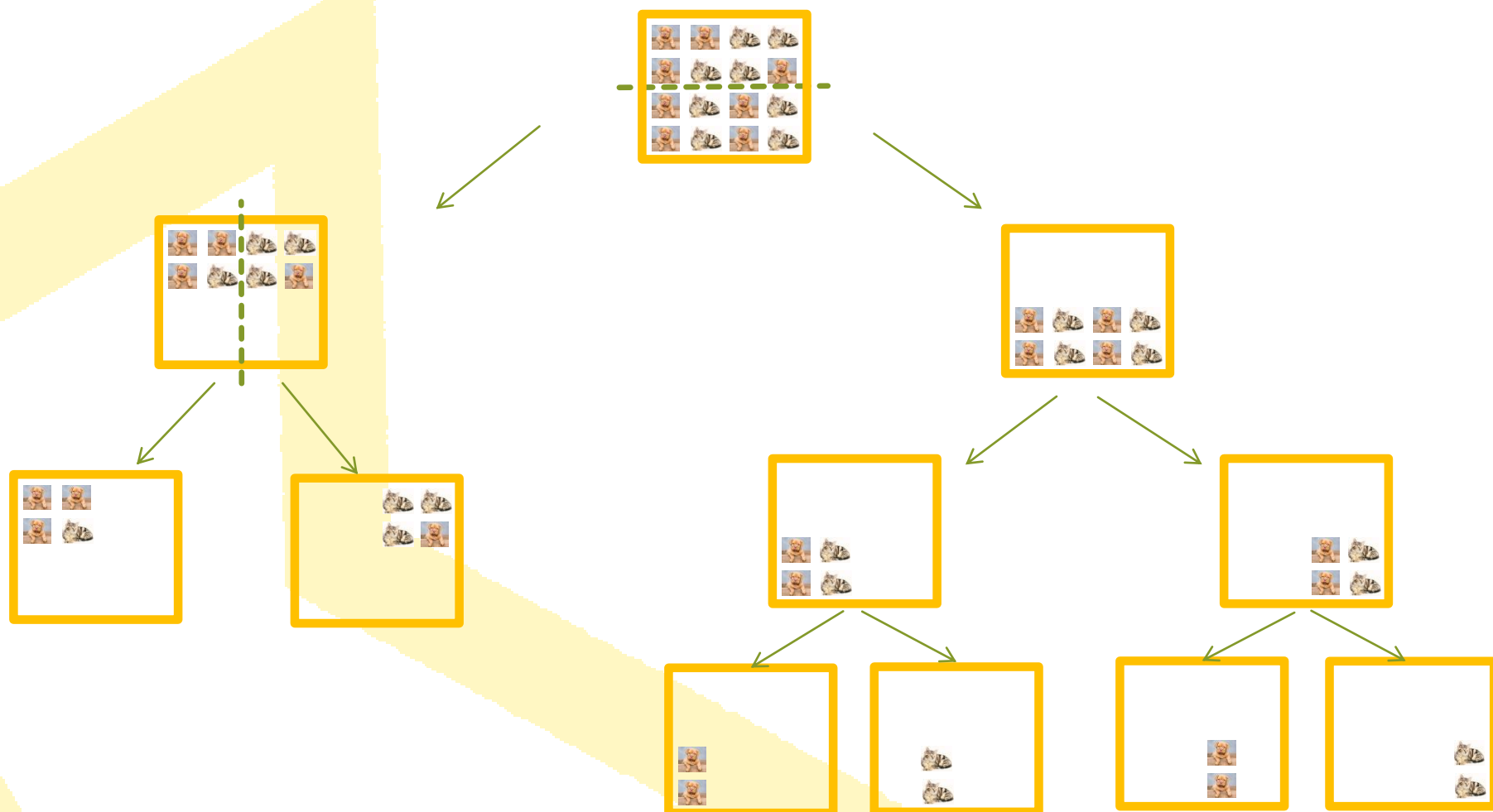
# Árbol de clasificación



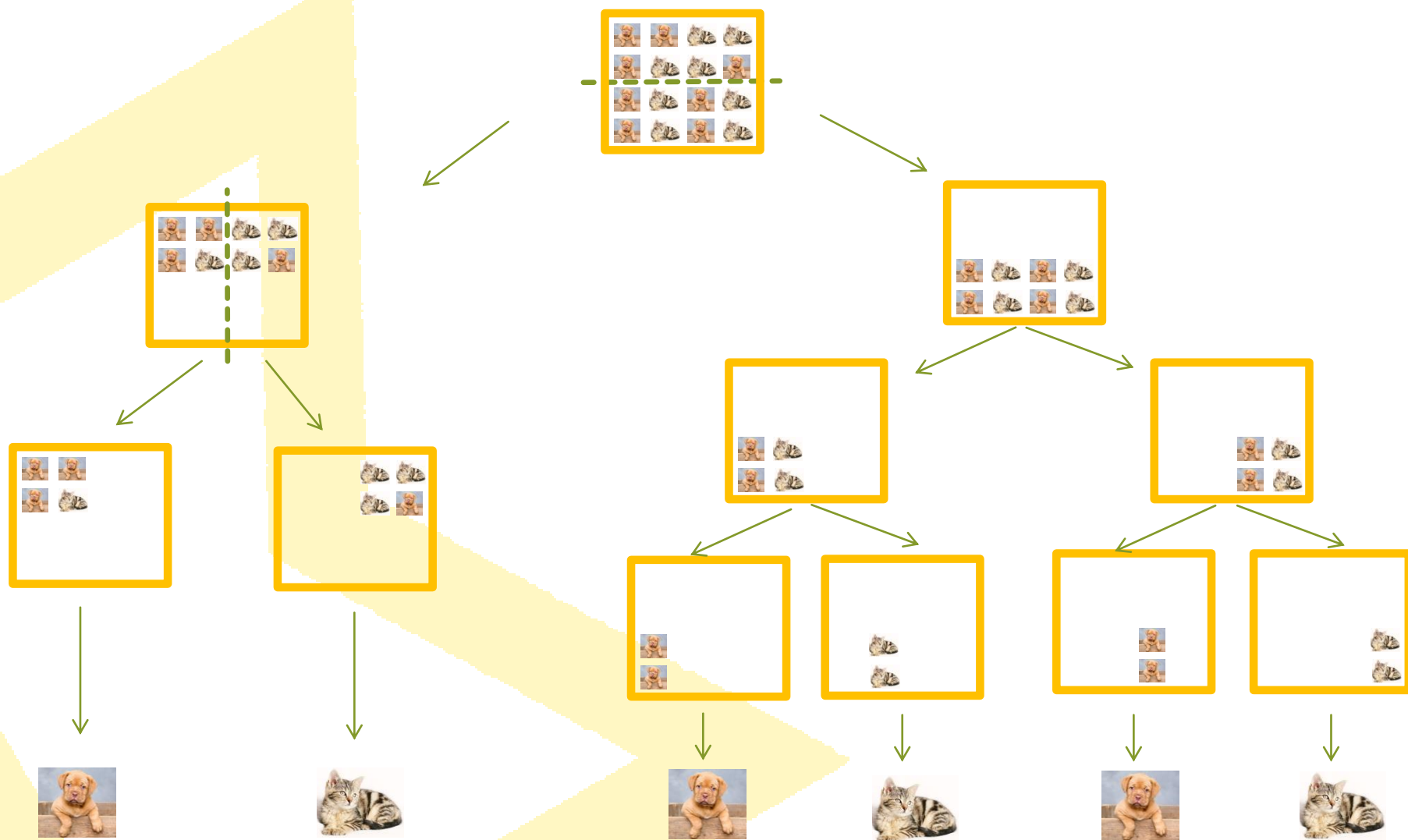
# Árbol de clasificación



# Árbol de clasificación



# Árbol de clasificación



## Agenda

2.1- Árbol de Clasificación

---

**2.2- Árboles de Regresión**

---

2.3- Criterios de creación de un árbol

---

2.4- Métricas

---

2.5- Criterio detener un árbol

---

2.6- Criterio de decidir la predicción

---

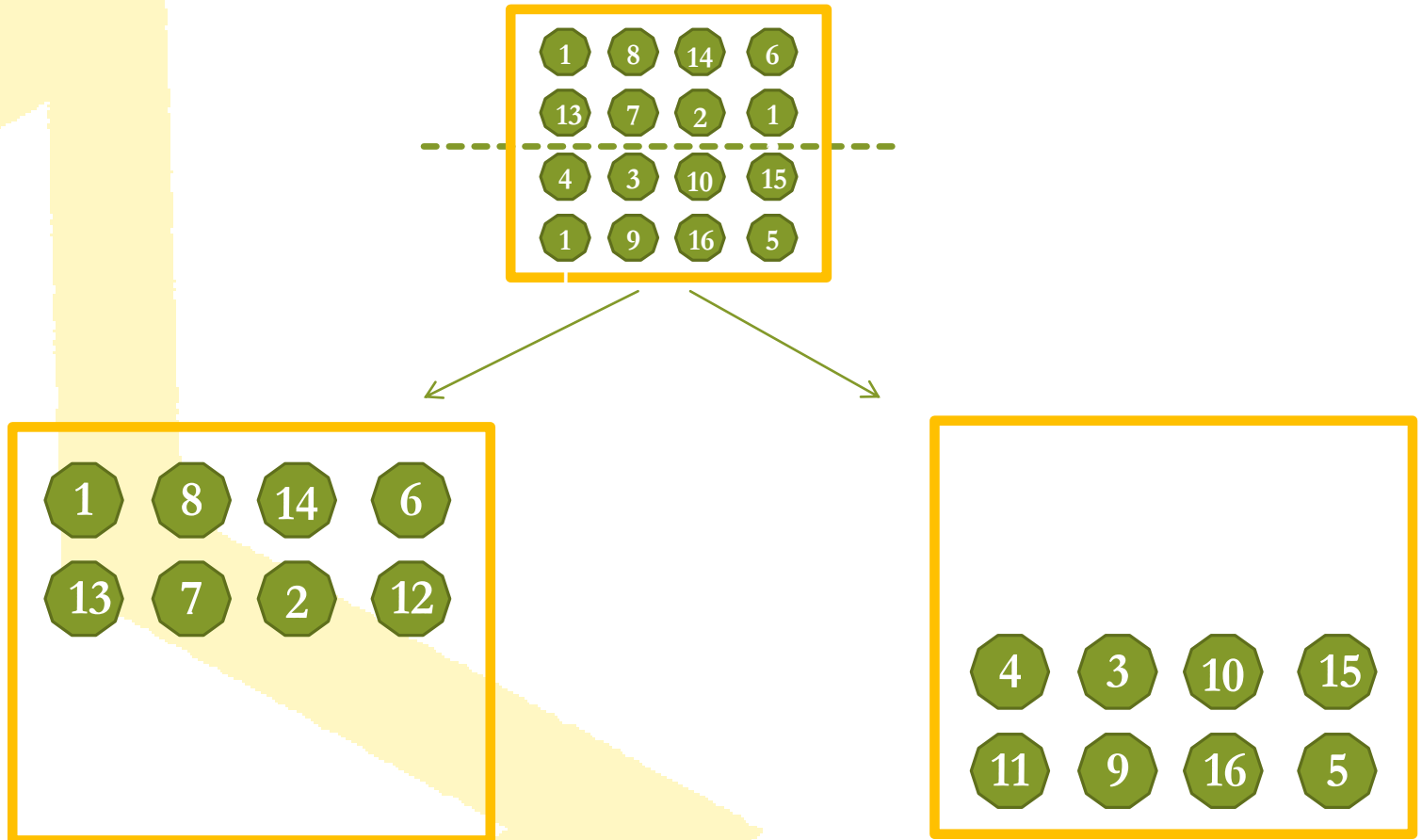
2.7- Cortes avanzados

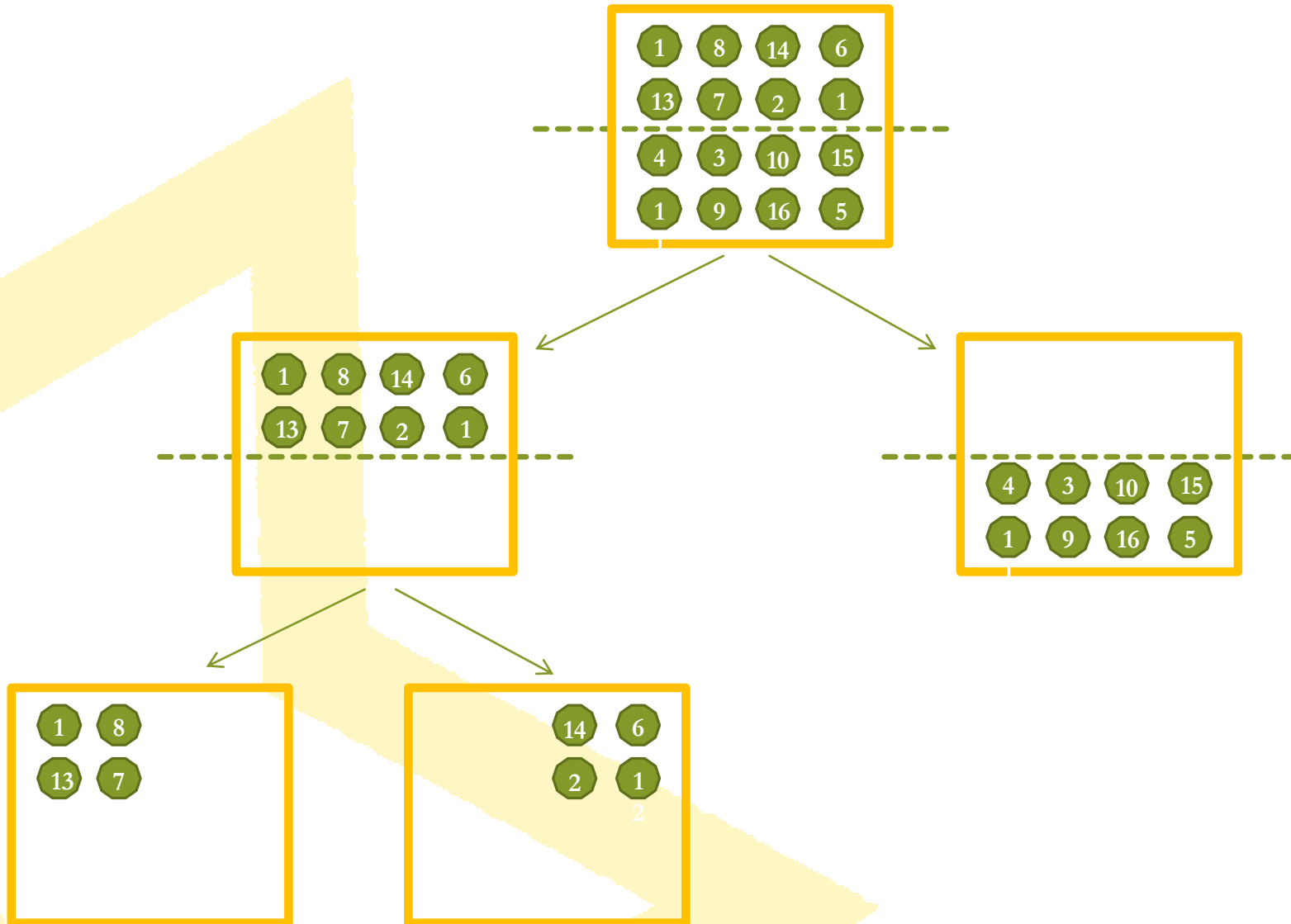


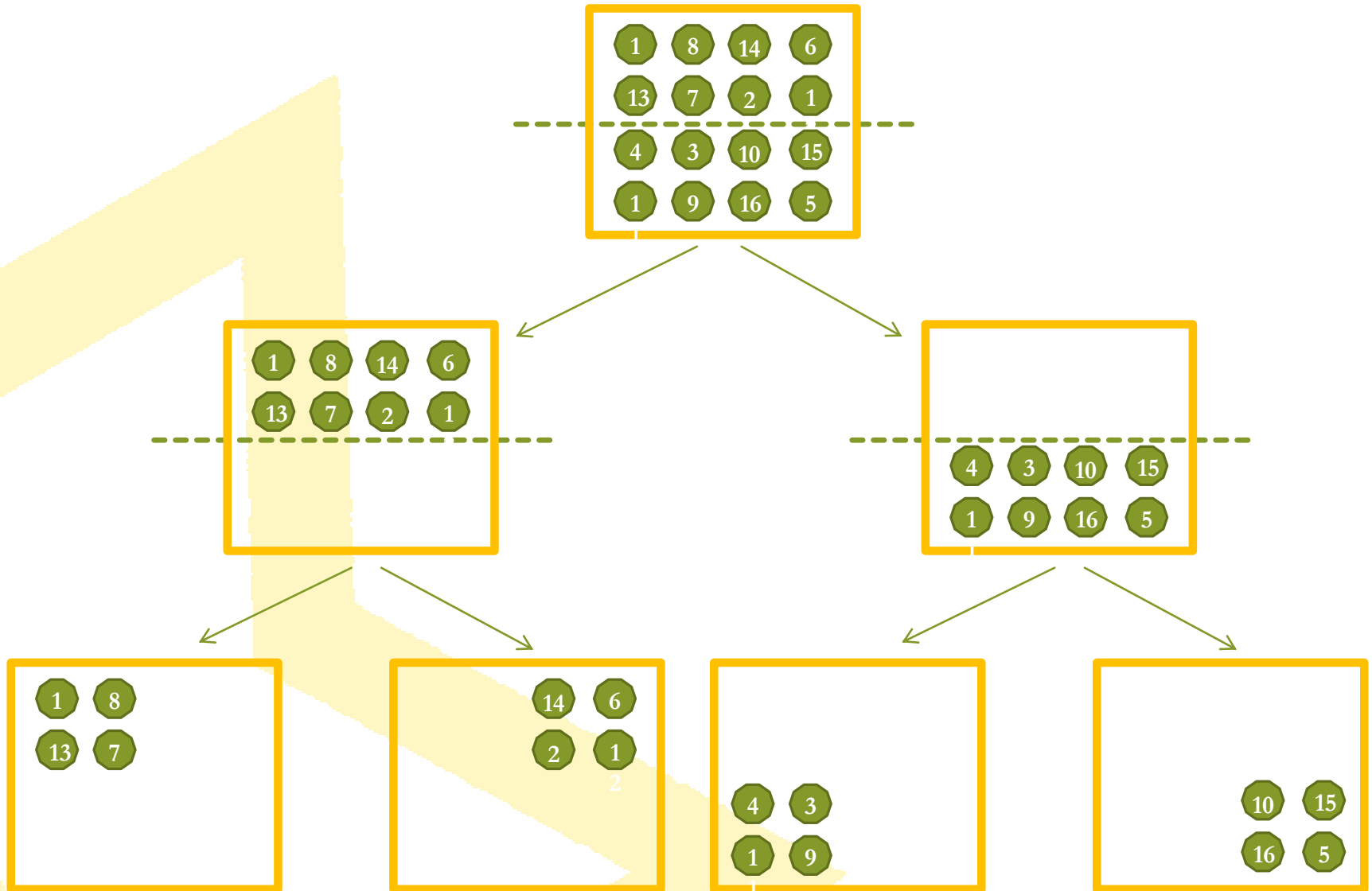
# Árbol de Regresión

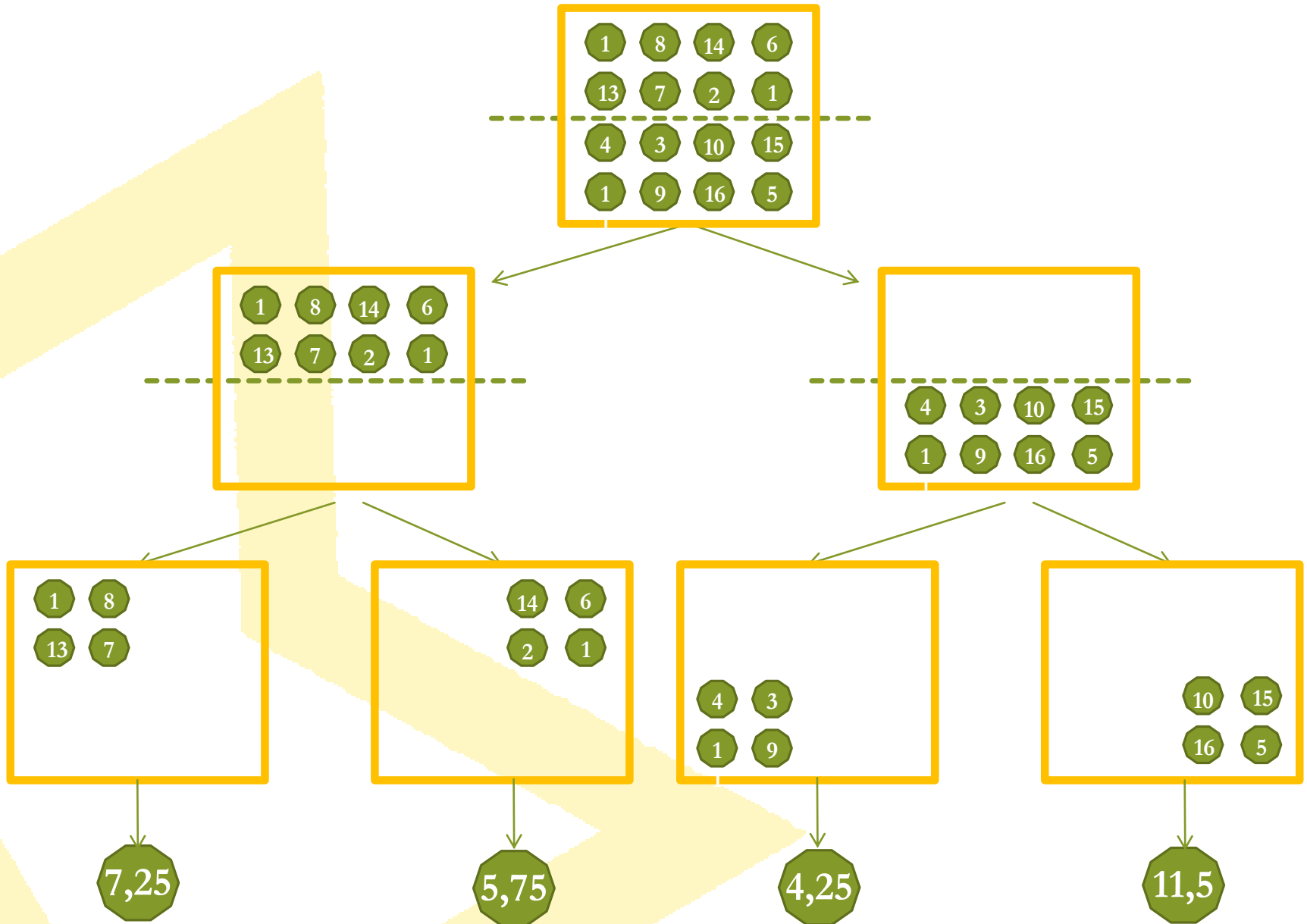


# Árbol de Regresión









## Agenda

2.1- Árbol de Clasificación

---

2.2- Árboles de Regresión

---

**2.3- Criterios de creación de un árbol**

---

2.4- Métricas

---

2.5- Criterio detener un árbol

---

2.6- Criterio de decidir la predicción

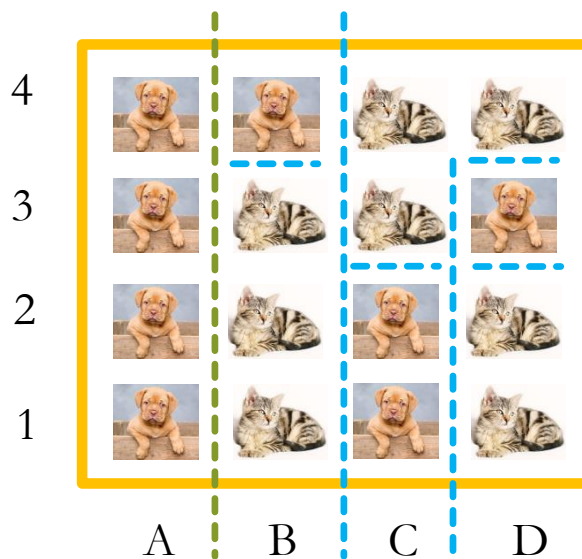
---

2.7- Cortes avanzados

# Criterios de creación del árbol



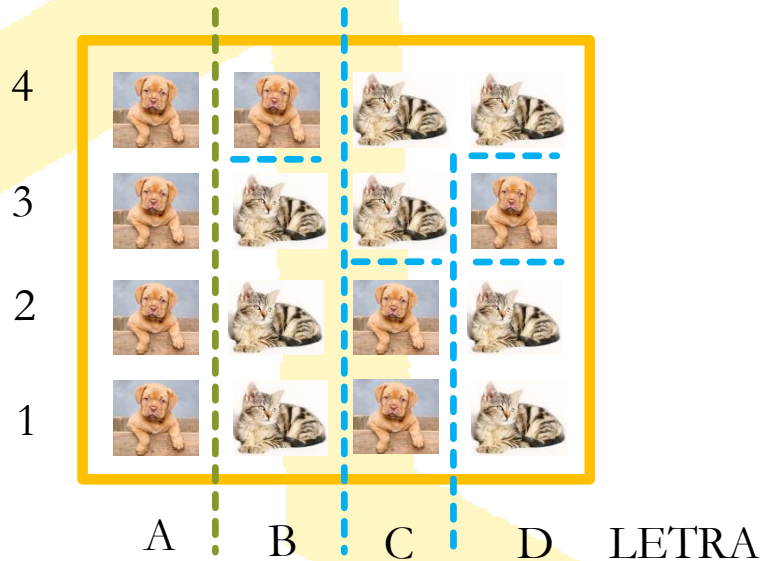
# Criterios de creación del árbol





# Criterios de creación del árbol

NÚMERO



LETRA = A

















LETRA = C and NÚMERO < 3

LETRA = D and NÚMERO = 3

LETRA = B and NÚMERO = 4

# Criterios de creación del árbol

NÚMERO

|   |   |   |   |   |
|---|---|---|---|---|
| 4 |  |  |  |  |
| 3 |  |  |  |  |
| 2 |  |  |  |  |
| 1 |  |  |  |  |
|   | A   | B   | C   | D   |

LETRA

LETRA = A

LETRA = C and NÚMERO < 3

LETRA = D and NÚMERO = 3

LETRA = B and NÚMERO = 4

AND

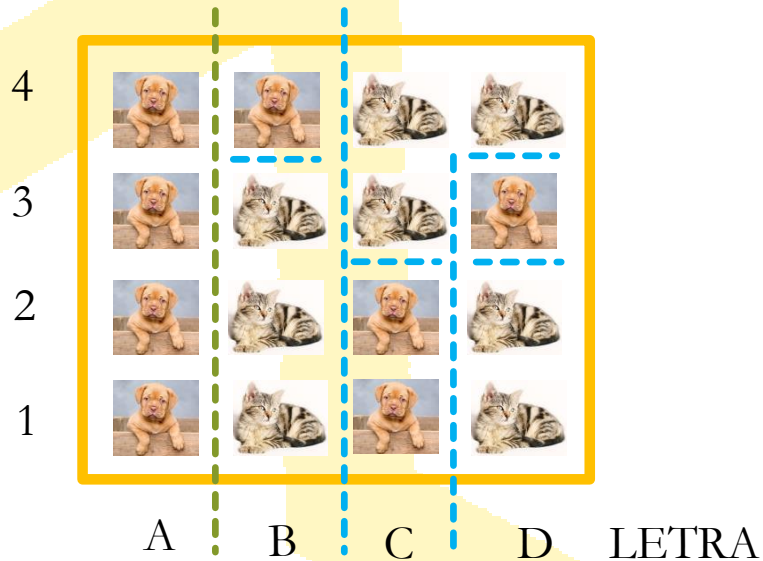
AND

AND



# Criterios de creación del árbol

NÚMERO



VARIABLE DISCRETA

LETRA = A

LETRA = B

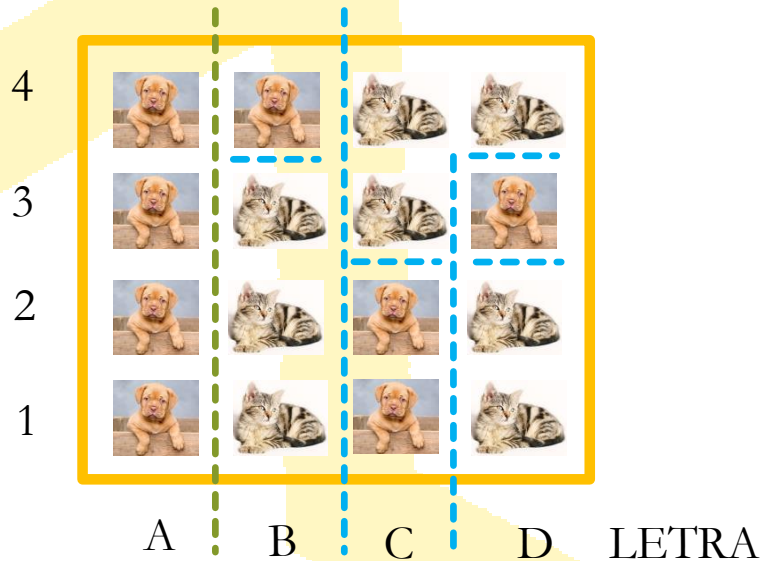
LETRA = C

LETRA = D

LETRA = A, LETRA = B, LETRA = C, LETRA = D

# Criterios de creación del árbol

NÚMERO



VARIABLE CONTINUA

$\text{NÚMERO} > 2$

$\text{NÚMERO} \leq 3$

$\text{NÚMERO} < 4$

$\text{NÚMERO} > 1$

$\text{NÚMERO1} < \text{NÚMERO2} < \text{NÚMERO3} < \text{NÚMERO4}$

## Agenda

2.1- Árbol de Clasificación

---

2.2- Árboles de Regresión

---

2.3- Criterios de creación de un árbol

---

**2.4- Métricas**

---

2.5- Criterio detener un árbol

---

2.6- Criterio de decidir la predicción

---

2.7- Cortes avanzados

# Métricas

Algoritmo de Clasificación:

| Matriz de Confusion |          | Predicho            |                      |   |           |
|---------------------|----------|---------------------|----------------------|---|-----------|
|                     |          | Negativo            | Positivo             |   |           |
| Real                | Negativo | a                   | b                    | <b>Verdadero Negativo (True negative rate)</b>  | $a/(a+b)$ |
|                     | Positivo | c                   | d                    | <b>Exactitud</b>                                | $d/(c+d)$ |
|                     |          | <b>Sensibilidad</b> | <b>Especificidad</b> | <b>Precisión = <math>(a+d)/(a+b+c+d)</math></b> |           |
|                     |          | $d/(d+c)$           | $a/(a+b)$            |   |           |

Figura 3: Matriz de confusión con otras métricas de evaluación.

El significado de cada uno de los términos es el siguiente:

- *a* es el número de predicciones correctas de clase negativa (negativos reales)
- *b* es el número de predicciones incorrectas de clase positiva (falsos positivos)
- *c* es el número de predicciones incorrectas de clase negativa (falsos negativos)
- *d* es el número de predicciones correctas de clase positiva (positivos reales)

Fuente: Luca

url: <https://empresas.blogthinkbig.com/ml-a-tu-alcance-matriz-confusion/>

# Métricas

Algoritmo de Regresión:

$$MAE = \frac{1}{n} \sum |y - \hat{y}|$$

Divide by the total number of data points

Predicted output value

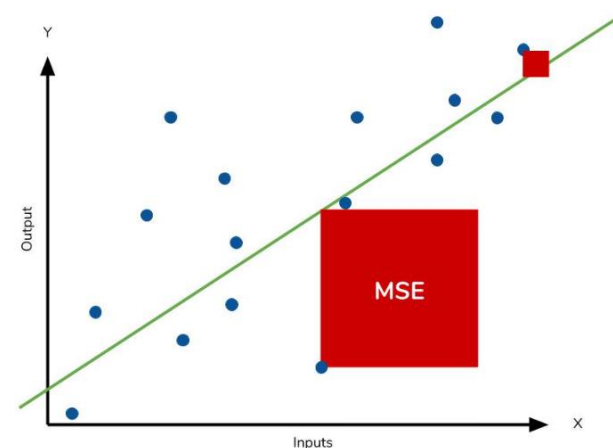
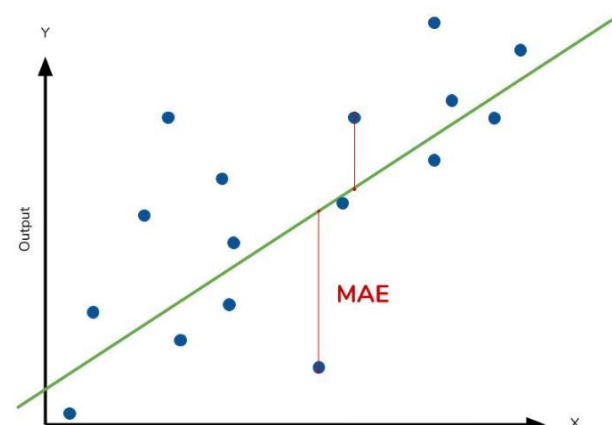
Actual output value

Sum of

The absolute value of the residual

$$MSE = \frac{1}{n} \sum (y - \hat{y})^2$$

The square of the difference between actual and predicted



Fuente: dataquest

url: <https://www.dataquest.io/blog/understanding-regression-error-metrics/>

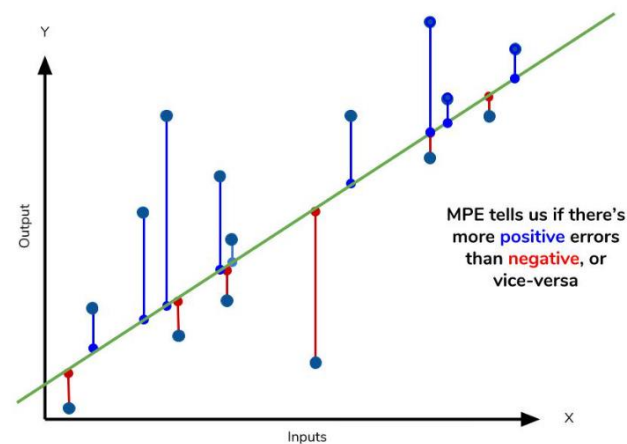
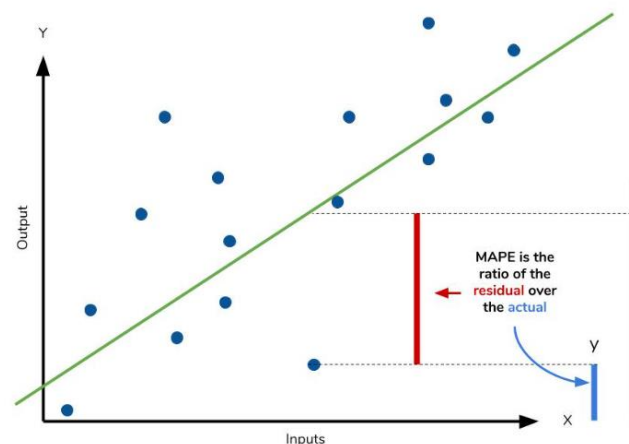
# Métricas

Algoritmo de Regresión:

$$MAPE = \frac{100\%}{n} \sum \left| \frac{\overbrace{y - \hat{y}}^{\text{The residual}}}{\underbrace{y}_{\text{Each residual is scaled against the actual value}}} \right|$$

Multiplying by 100% converts to percentage

$$MPE = \frac{100\%}{n} \sum \left( \frac{y - \hat{y}}{y} \right)$$



Fuente: dataquest

url: <https://www.dataquest.io/blog/understanding-regression-error-metrics/>



## Agenda

2.1- Árbol de Clasificación

---

2.2- Árboles de Regresión

---

2.3- Criterios de creación de un árbol

---

2.4- Métricas

---

**2.5- Criterio detener un árbol**

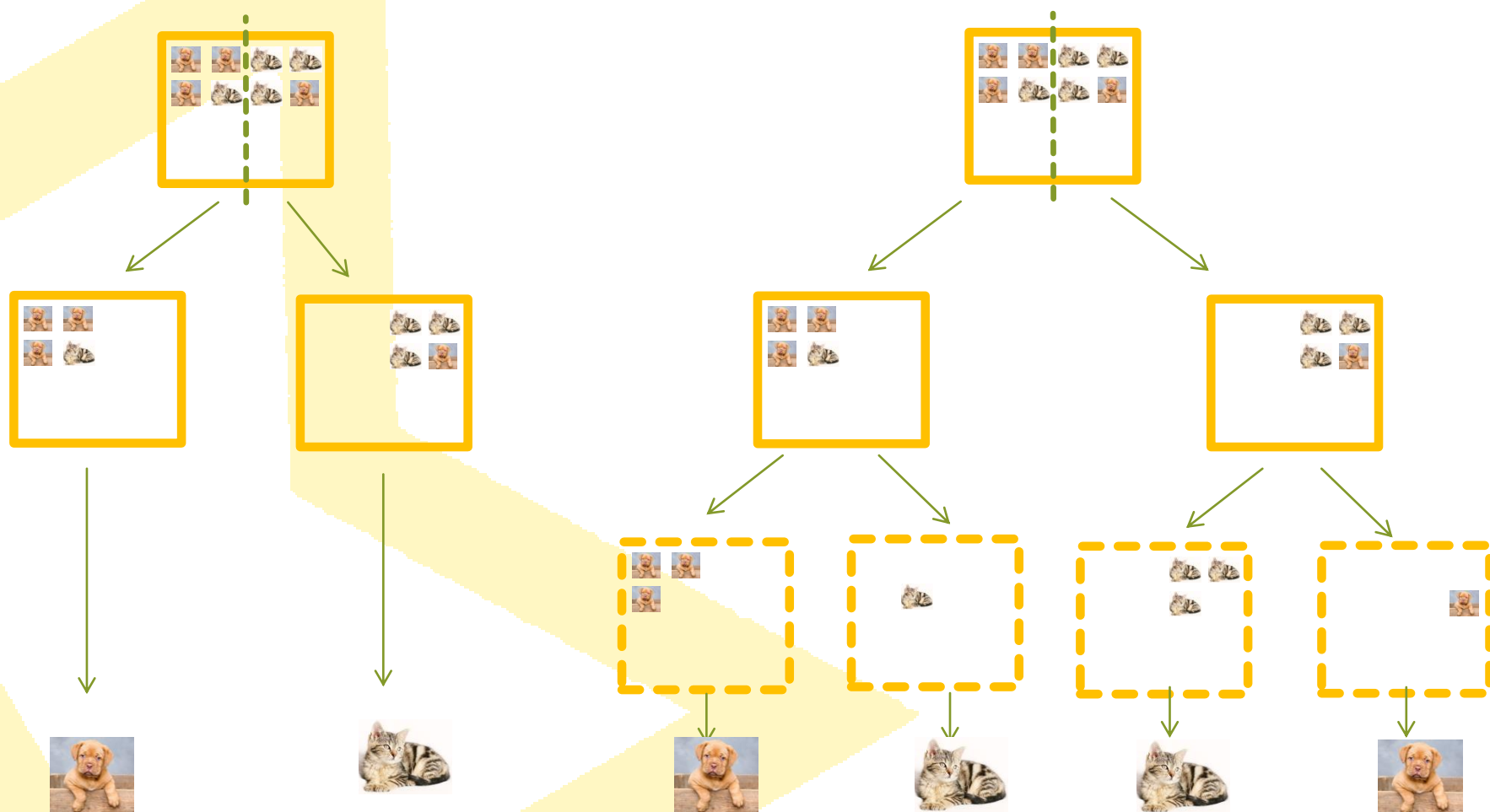
---

2.6- Criterio de decidir la predicción

---

2.7- Cortes avanzados

# Criterio para detener la construcción del árbol



## Agenda

2.1- Árbol de Clasificación

---

2.2- Árboles de Regresión

---

2.3- Criterios de creación de un árbol

---

2.4- Métricas

---

2.5- Criterio detener un árbol

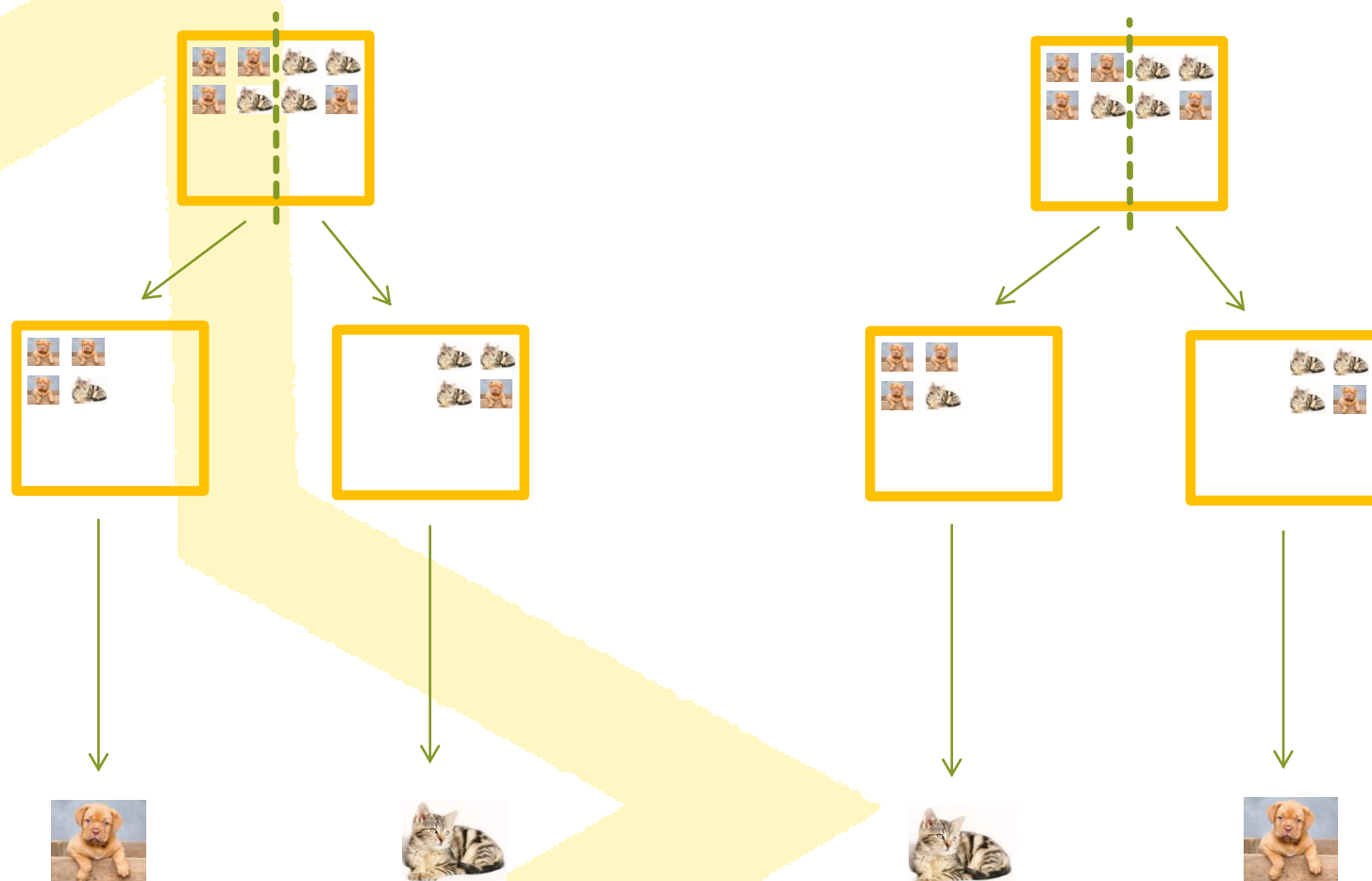
---

**2.6- Criterio de decidir la predicción**

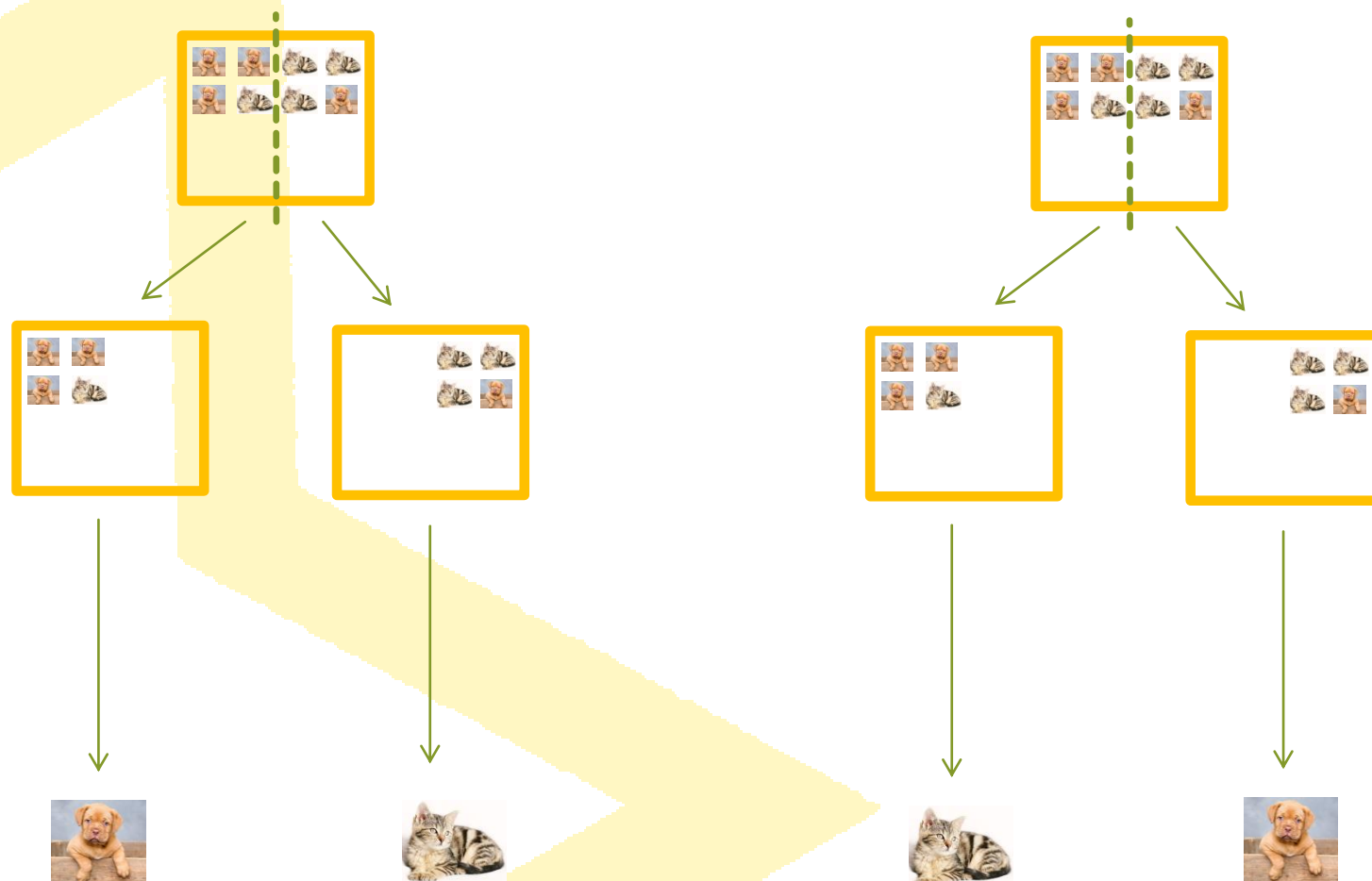
---

2.7- Cortes avanzados

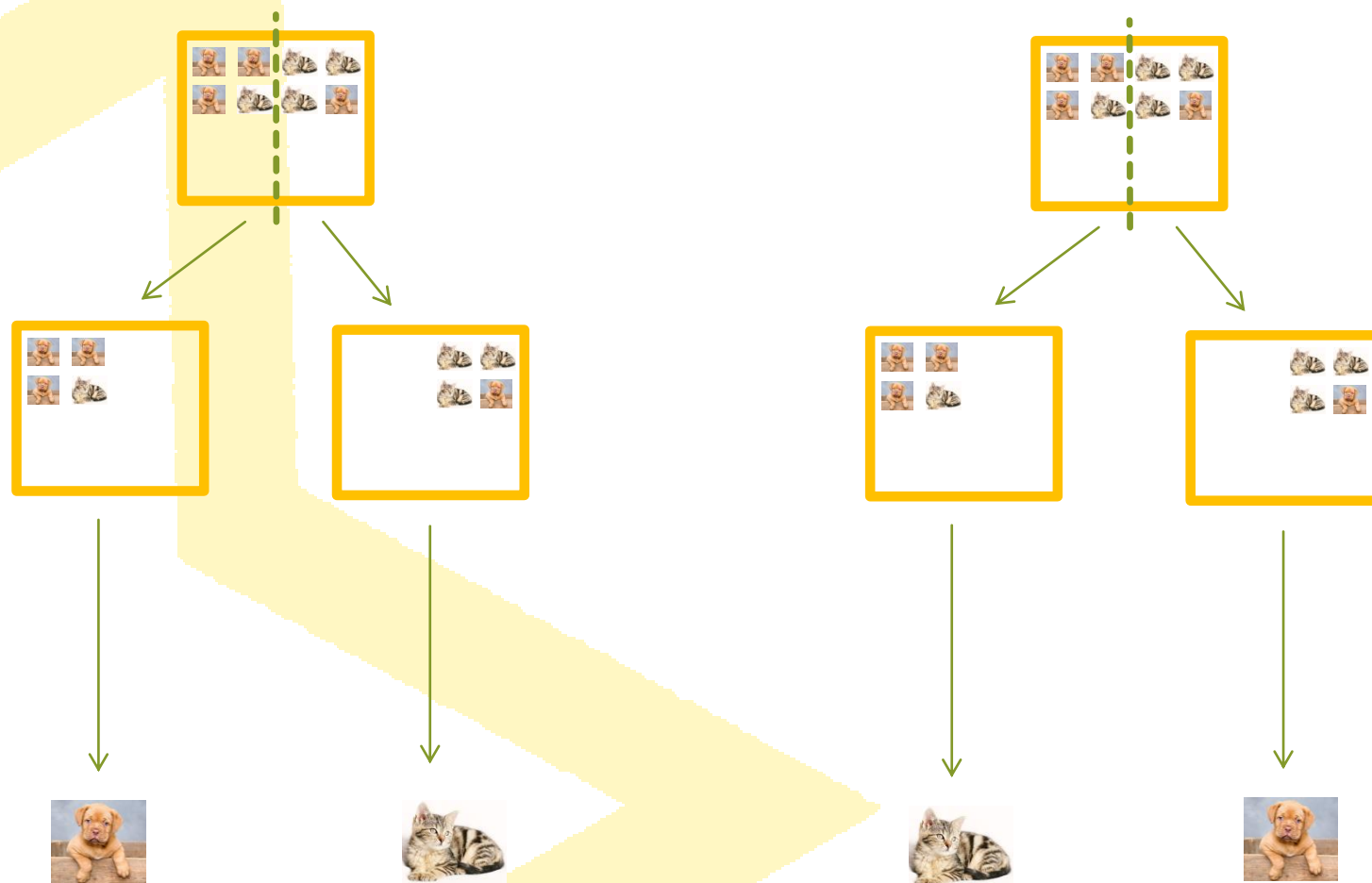
# Criterio para decidir la predicción



# Ejercicio de predicción 1



## Ejercicio de predicción 2



## Agenda

2.1- Árbol de Clasificación

---

2.2- Árboles de Regresión

---

2.3- Criterios de creación de un árbol

---

2.4- Métricas

---

2.5- Criterio detener un árbol

---

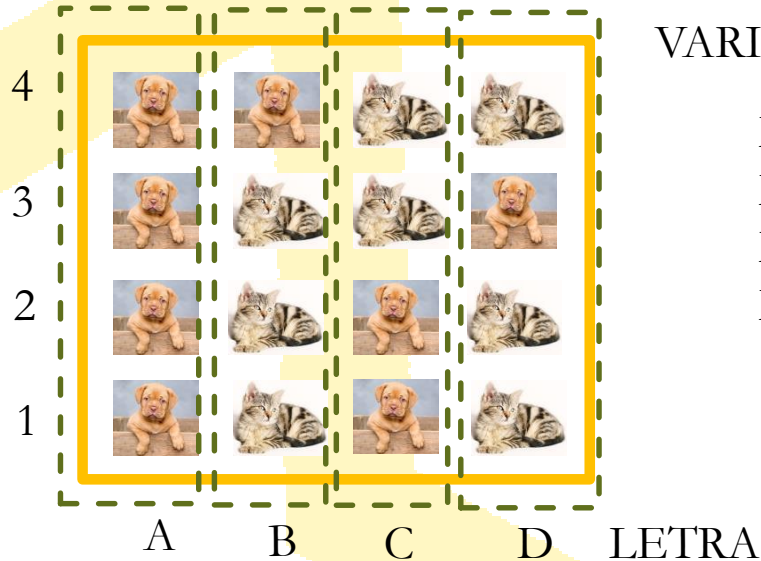
2.6- Criterio de decidir la predicción

---

**2.7- Cortes avanzados**

# Cortes avanzados

NÚMERO



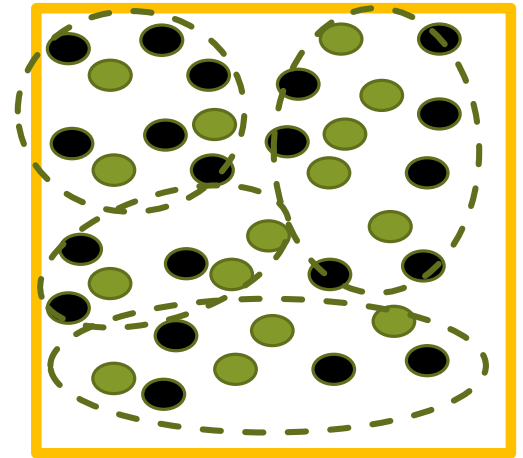
VARIABLE DISCRETA

LETRA = A

LETRA = B

LETRA = C

LETRA = D



LETRA = A, LETRA = B, LETRA = C, LETRA = D



## Cortes avanzados

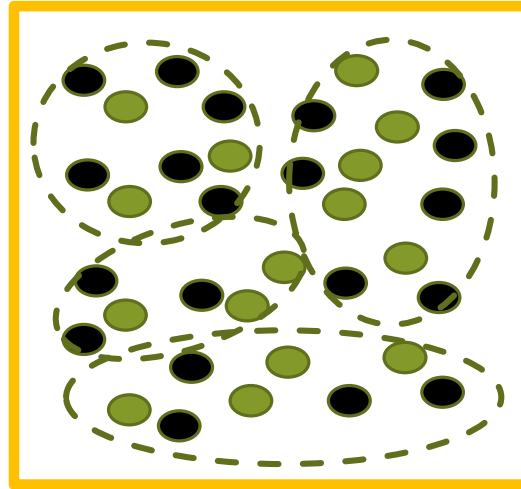
VARIABLE DISCRETA

LETRA = A

LETRA = B

LETRA = C

LETRA = D

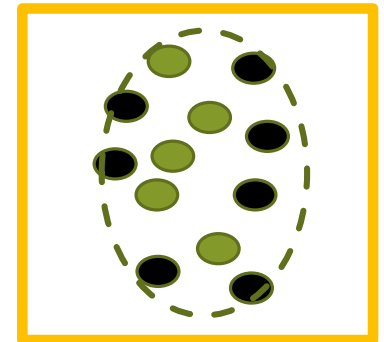
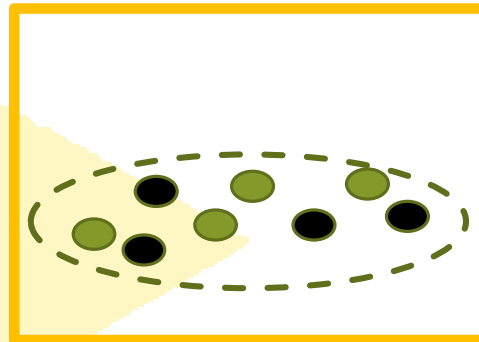
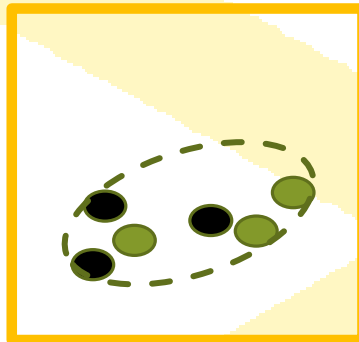
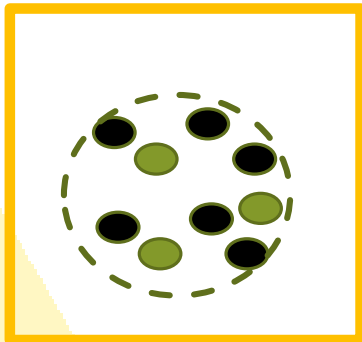


A

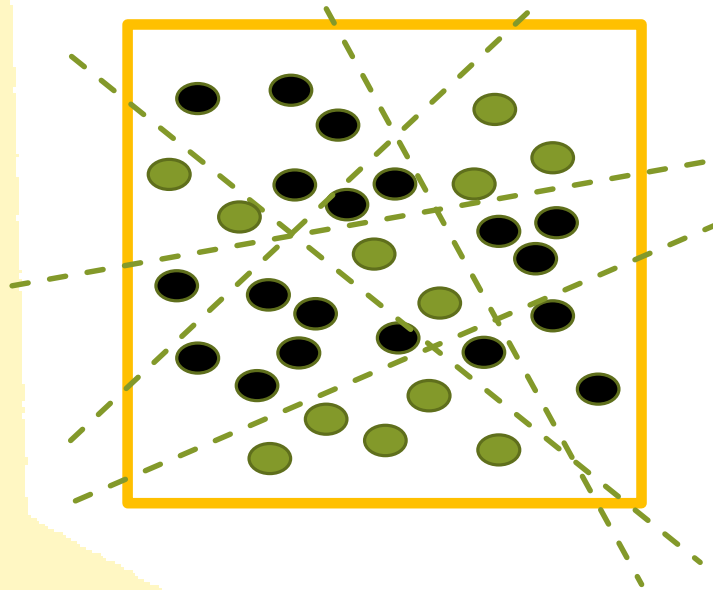
B

C

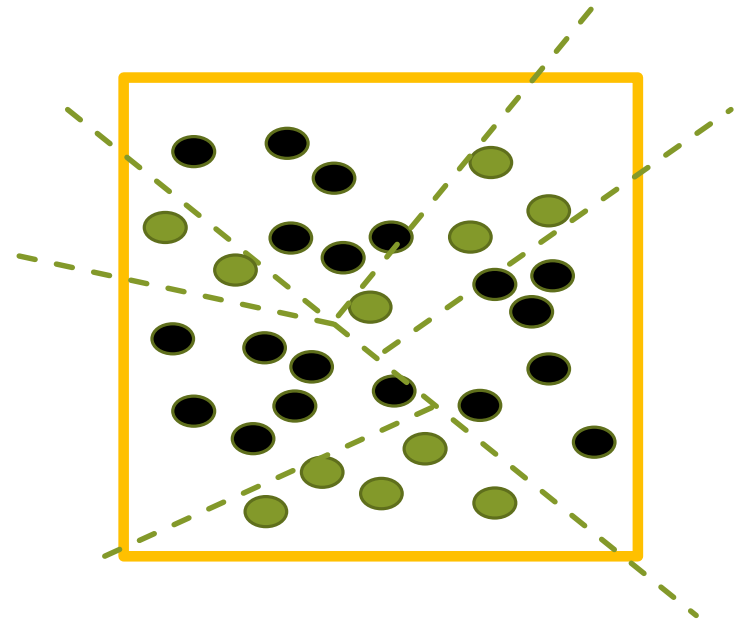
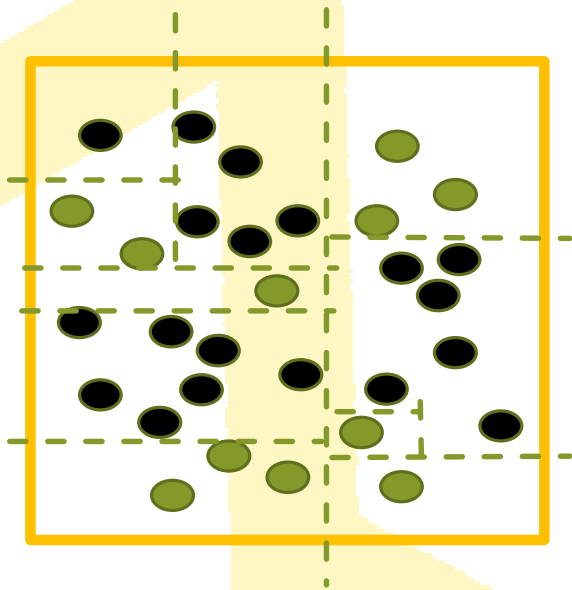
D



# Cortes oblicuos



# Cortes oblicuos



## Agenda

1- Clasificación vs Regresión

---

2- Árboles de decisión

---

**3- Bosques**

---

4- K-vecinos

---

4- Ensembles

---

6- Fases de un proyecto

---

7- Ejercicios guiados

# Bosques

RANDOM  
FOREST

=

PASTING  
BAGGING

# Bosques

- Tanto en bagging como pasting se combinan diferentes versiones del mismo modelo, el cual es entrenado con diferentes conjuntos de datos.
- Uno de los modelos más populares de este tipo de modelos es el Random Forest.

# Bosques

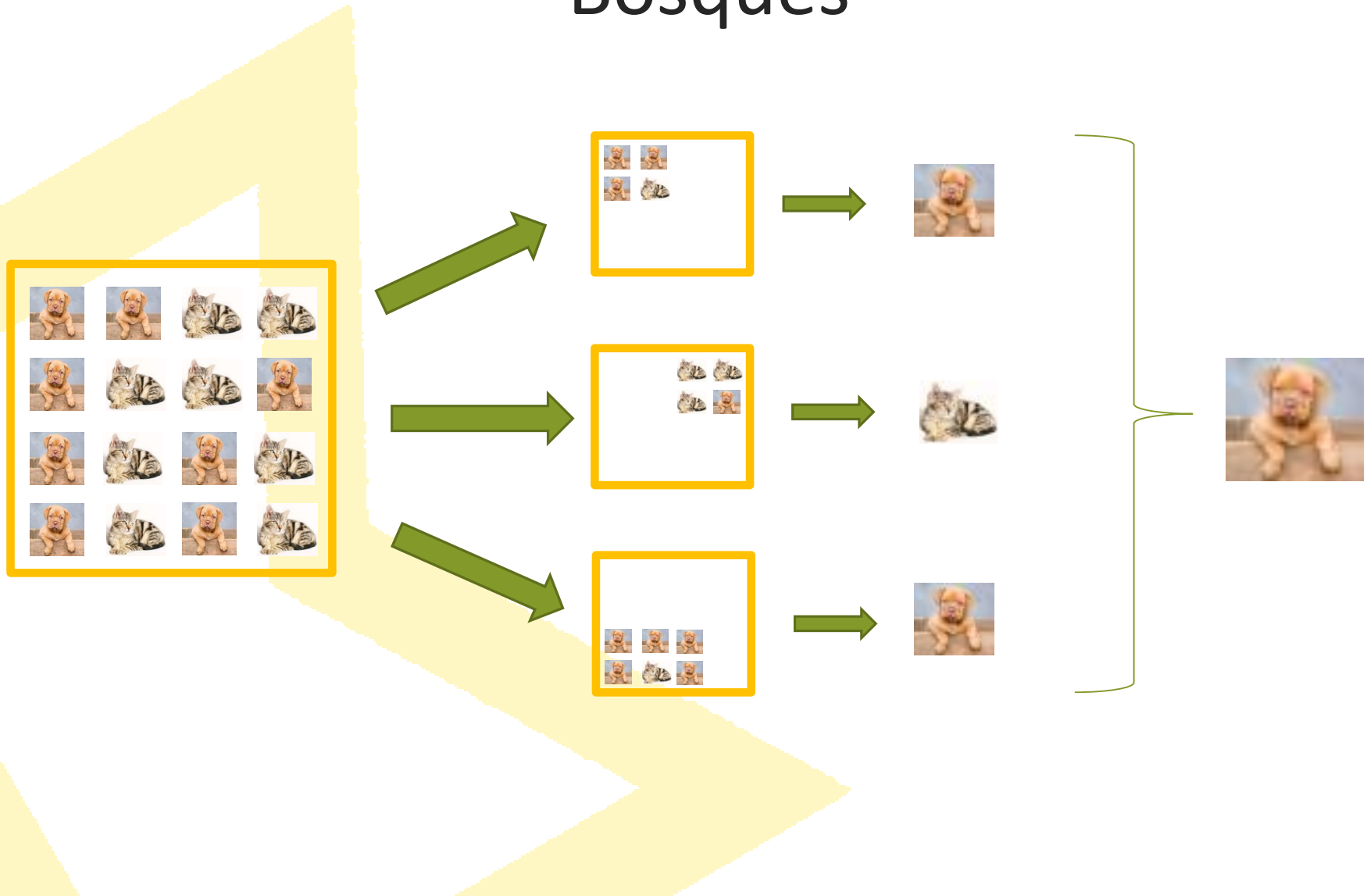
## PASTING

- Cuando selecciona un registro para entrenar este **NO** se vuelve a introducir en la lista de candidatos, es decir, un registro puede **NO** aparecer más de una vez en el conjunto de datos.

## BAGGING

- Cuando selecciona un registro para entrenar este se vuelve a introducir en la lista de candidatos, es decir, un registro puede aparecer más de una vez en el conjunto de datos.

## Bosques





## Agenda

1- Clasificación vs Regresión

---

2- Árboles de decisión

---

3- Bosques

---

**4- K-vecinos**

---

5- Ensembles

---

6- Tipos de datos en Big Data

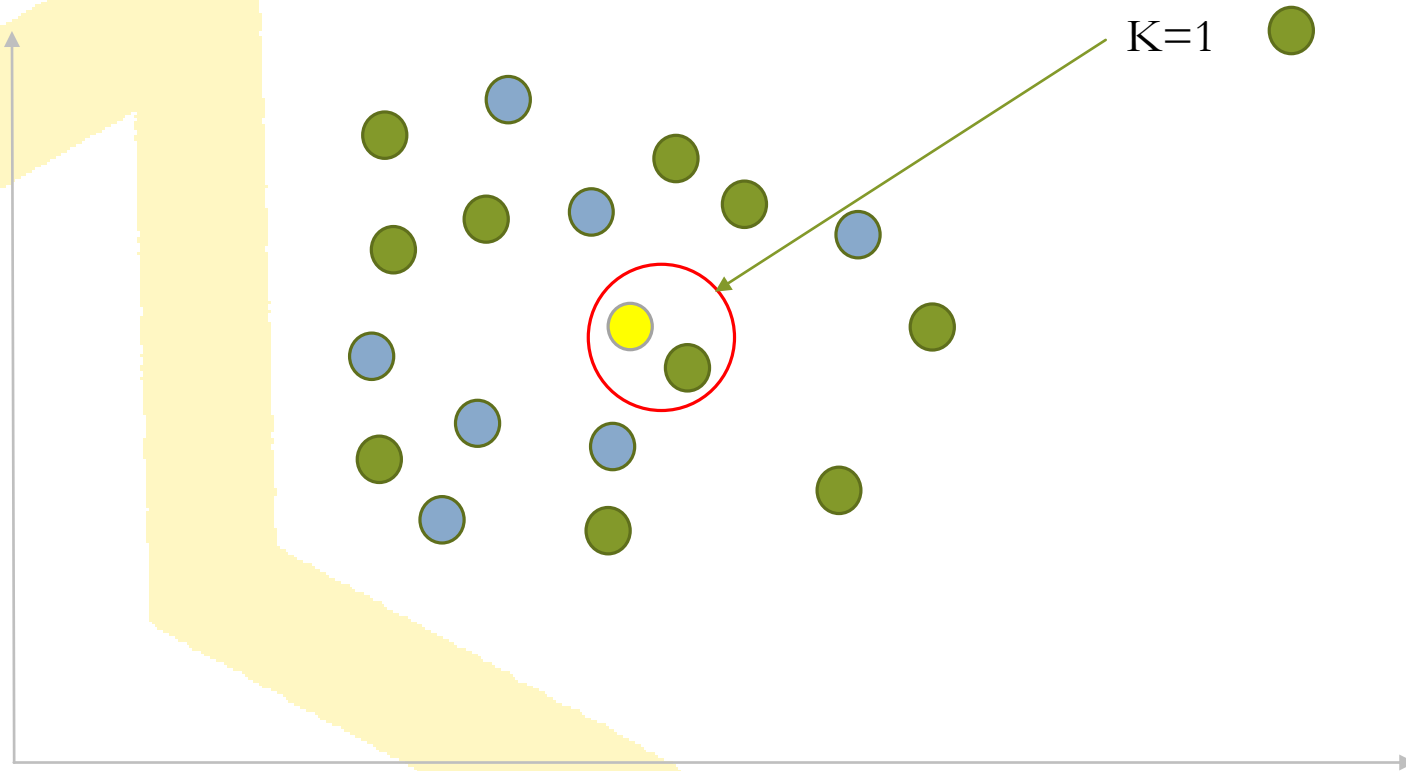
---

7- Herramientas para aplicar Data Driven Marketing

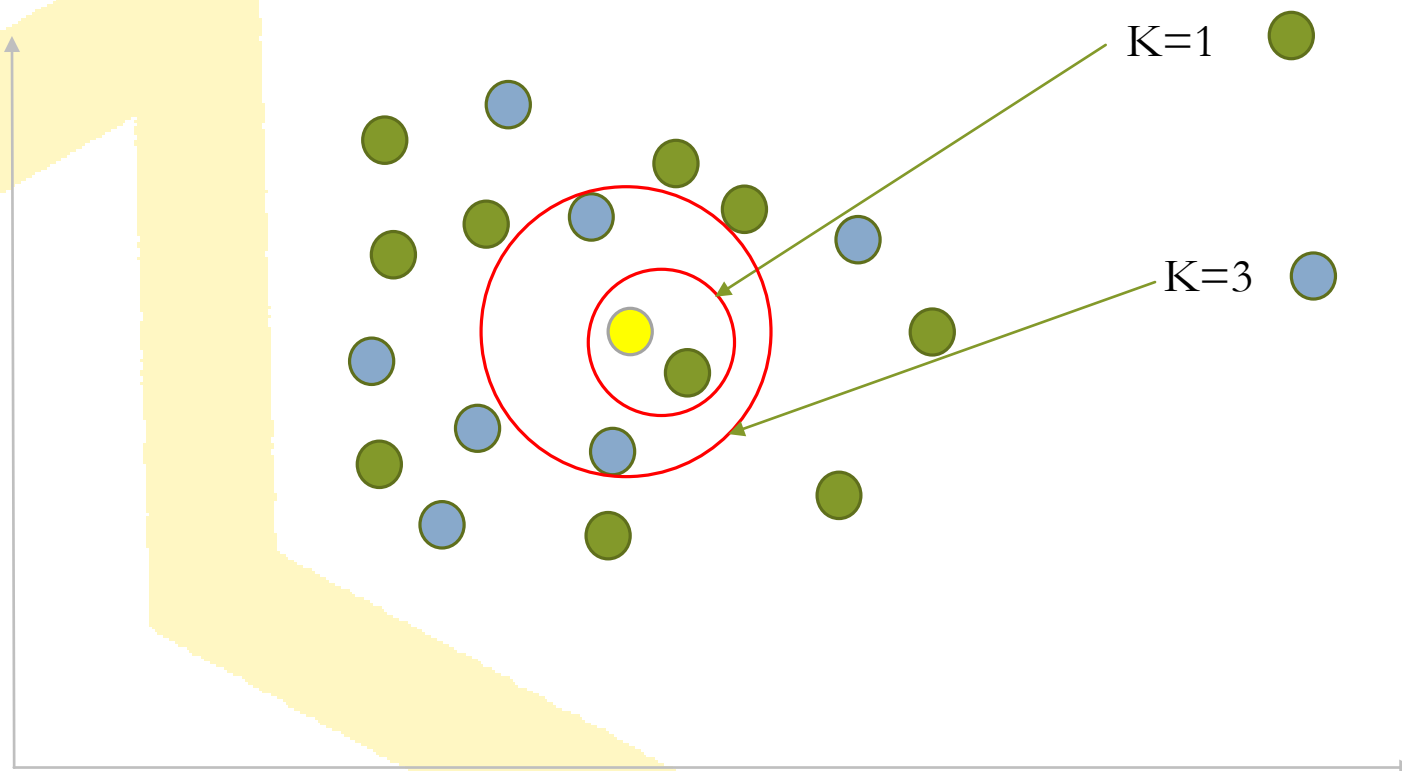
## K-vecinos

- Es un método de clasificación supervisada, y se basa en estimar la función de densidad.
- El algoritmo k-nn, es usado para la clasificación de registros basado en un entrenamiento mediante ejemplos cercanos en el espacio. Por ello, el método no realiza suposiciones sobre las variables predictoras.

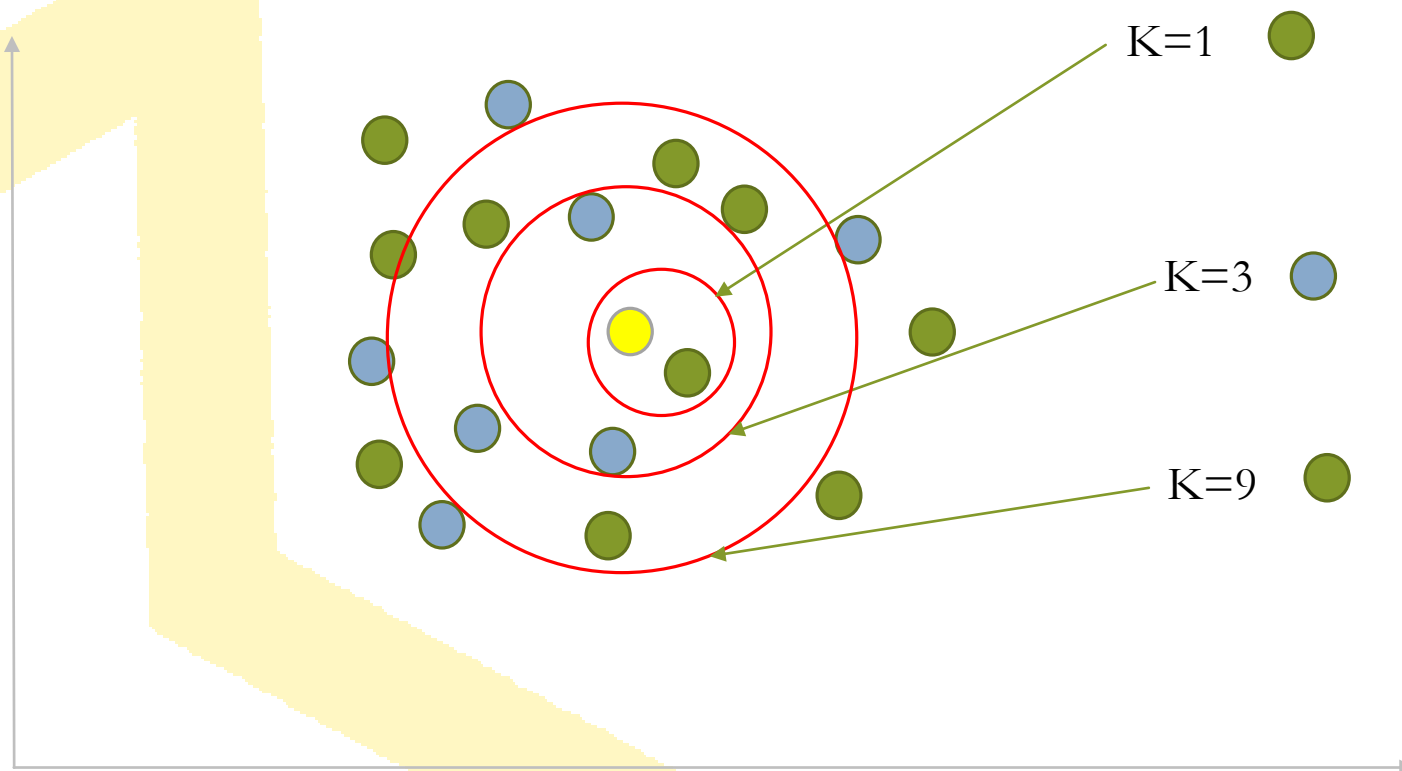
## K-vecinos



## K-vecinos



## K-vecinos



## ¿Cómo elegimos el k?

- Depende de los datos y del proyecto.
- De forma general:  $k=\sqrt{n}$
- Valores muy grandes de k reducen el ruido, pero crean límites muy grandes, perdiendo la capacidad de definir pequeños grupos con comportamiento similar.
- Valores muy pequeños por el contrario, definen grupos con mucho detalle, pero son muy sensibles al ruido.

## Agenda

1- Clasificación vs Regresión

---

2- Árboles de decisión

---

3- Bosques

---

4- K-vecinos

---

**5- Ensembles**

---

6- Fases de un proyecto

---

7- Ejercicios guiados

# Ensembles

- Combinar diferentes algoritmos para aprovechar y aglutinar las bondades de cada uno de ellos.
- De forma agregada, los métodos para construir ensembles son:
  - Bagging/Pasting
  - Boosting
  - Blending/Stacking



# Ensembles

## PASTING

- Cuando selecciona un registro para entrenar este **NO** se vuelve a introducir en la lista de candidatos, es decir, un registro puede **NO** aparecer más de una vez en el conjunto de datos.

## BAGGING

- Cuando selecciona un registro para entrenar este se vuelve a introducir en la lista de candidatos, es decir, un registro puede aparecer más de una vez en el conjunto de datos.

# Ensembles

## BOOSTING

- Se entrenan diferentes algoritmos sobre diferentes set de datos.
- La diferencia principal con los anteriores, es que este tipo entrena con los registros en los que otros algoritmos han fallado.

# Ensembles

## BLENDING/STACKING

- Se entrenan diferentes algoritmos sobre un mismo set de datos.
- Este tipo de ensembles no combina las predicciones parciales mediante una función de agregación, se utilizan esas predicciones como entradas de un nuevo modelo.

## Agenda

1- Clasificación vs Regresión

---

2- Árboles de decisión

---

3- Bosques

---

4- K-vecinos

---

5- Ensembles

---

**6- Fases de un proyecto**

---

7- Ejercicios guiados

## Agenda

**6.1- Definir el problema**

---

6.2- Preparar el dato

---

6.3- Check de los algoritmos

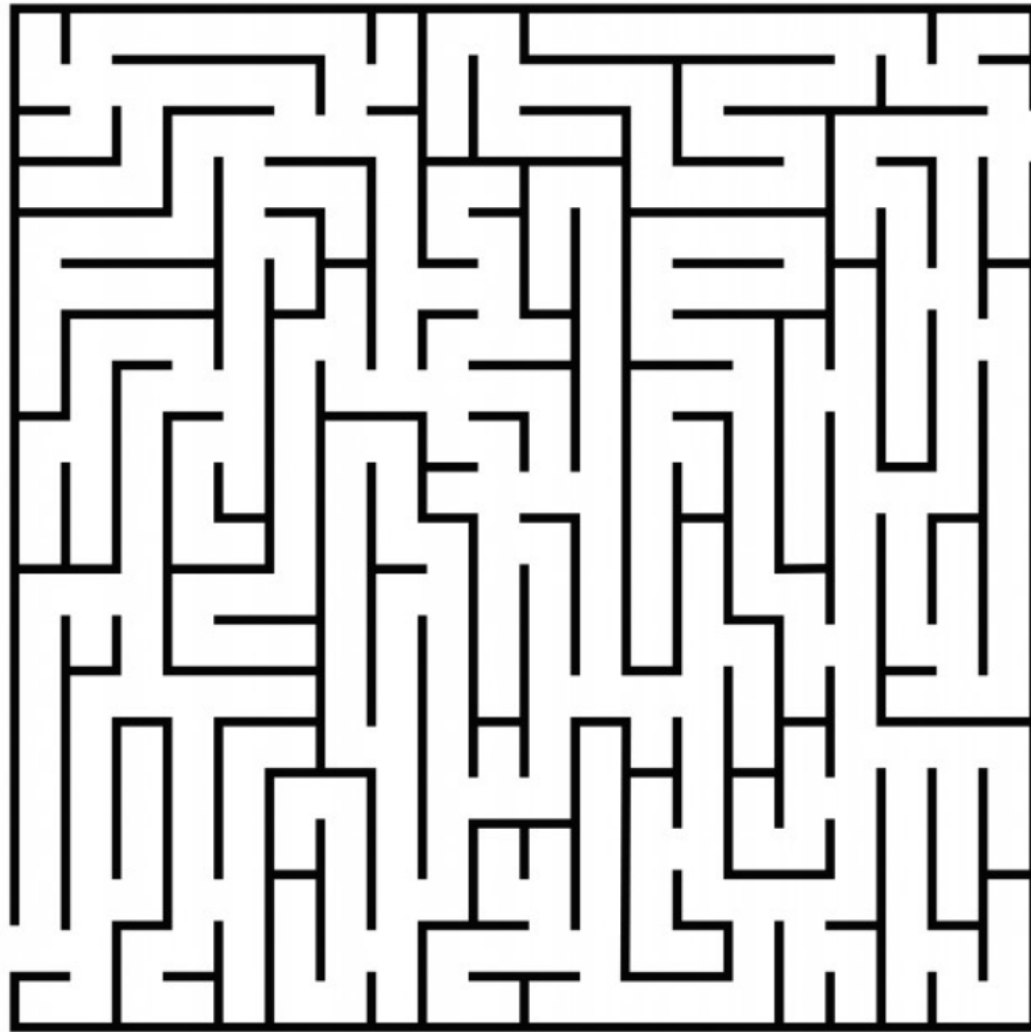
---

6.4- Mejorar los algoritmos

---

6.5- Presentar los resultados

---



# Definir el problema

- Es muy importante definir bien el problema, pero sobre todo, el objetivo de negocio que se quiere cumplir.
- Una vez hecho esto, se comunica a todos los implicados: Data science, Negocio, Marketing, IT, etc...
- Objetivo común = Misma dirección

# Definir el problema

- Dificultades:
  - Entender los requisitos de negocio
  - Definir bien el problema. No es posible desarrollar un proyecto que solucione todos los problemas!
  - Métrica
  - ¿Qué consideramos éxito?



## Agenda

6.1- Definir el problema

---

**6.2- Preparar el dato**

---

6.3- Check de los algoritmos

---

6.4- Mejorar los algoritmos

---

6.5- Presentar los resultados

---

## Preparar el dato

- Esta es la parte que más recursos consume en un proyecto, en torno al 80%.
- Se debe analizar el dato en profundidad, para conocer muy bien la calidad del dato y la limpieza que será necesario.
- Se debe recopilar todo el dato posible, y asegurar el acceso para las siguientes fases.
- ¡OJO! Este dato debe ser actualizado en el futuro.

# Preparar el dato

- Esta parte se puede dividir en 3 grandes áreas:
  - **Selección del dato:** Revisión de todo el dato disponible y recopilación para su uso en fases posteriores.
  - **Preprocesado del dato:** exploración y limpieza de los datos.
  - **Transformación del dato:** transformación de los datos preprocesados en la fase anterior, para poderlo utilizar directamente en los modelos.

## Agenda

6.1- Definir el problema

---

6.2- Preparar el dato

---

**6.3- Check de los algoritmos**

---

6.4- Mejorar los algoritmos

---

6.5- Presentar los resultados

---

## Check de los algoritmos

- Una vez que tenemos todos los datos preparados, podemos hacer la primera prueba de modelos.
- Hay que probar varios, y medir con la métrica fijada.
- ¿Cuánto tiempo debemos tardar en construir en esta fase una prueba de concepto?

## Check de los algoritmos

- Una vez que tenemos todos los datos preparados, podemos hacer la primera prueba de modelos.
- Hay que probar varios, y medir con la métrica fijada.
- ¿Cuánto tiempo debemos tardar en construir en esta fase una prueba de concepto?

# Check de los algoritmos

- ¿Cómo probamos los algoritmos?
  - En primer lugar, separar la muestra en dos dataset:

**TRAIN**

**70%**

**80%**

**TEST**

**30%**

**20%**

# Check de los algoritmos

- Entrenamos todos los algoritmos con el mismo conjunto de datos:

A large, solid orange rectangular button with the text '¡TRAIN!' in a white, bold, serif font, centered within the button.

¡TRAIN!

- Es **MUY IMPORTANTE** que todos los algoritmos entrenen con el mismo data set, sino no podrán ser comparables.



# Check de los algoritmos

- Una vez entrenados todos los algoritmos, los probamos contra otro conjunto de datos:

**¡TEST!**

- Evaluamos nuestros modelos realizando predicciones con los registros del dataset de Test, con la métrica fijada anteriormente.

## Agenda

6.1- Definir el problema

---

6.2- Preparar el dato

---

6.3- Check de los algoritmos

---

**6.4- Mejorar los algoritmos**

---

6.5- Presentar los resultados

---

# Mejorar los algoritmos

- Revisión y mejora del set de datos
- Optimización de los parámetros configurados en el algoritmo al entrenarlo.
- Composición de output en base a diferentes algoritmos para combinar la bondad de los modelos.

## Agenda

6.1- Definir el problema

---

6.2- Preparar el dato

---

6.3- Check de los algoritmos

---

6.4- Mejorar los algoritmos

---

**6.5- Presentar los resultados**

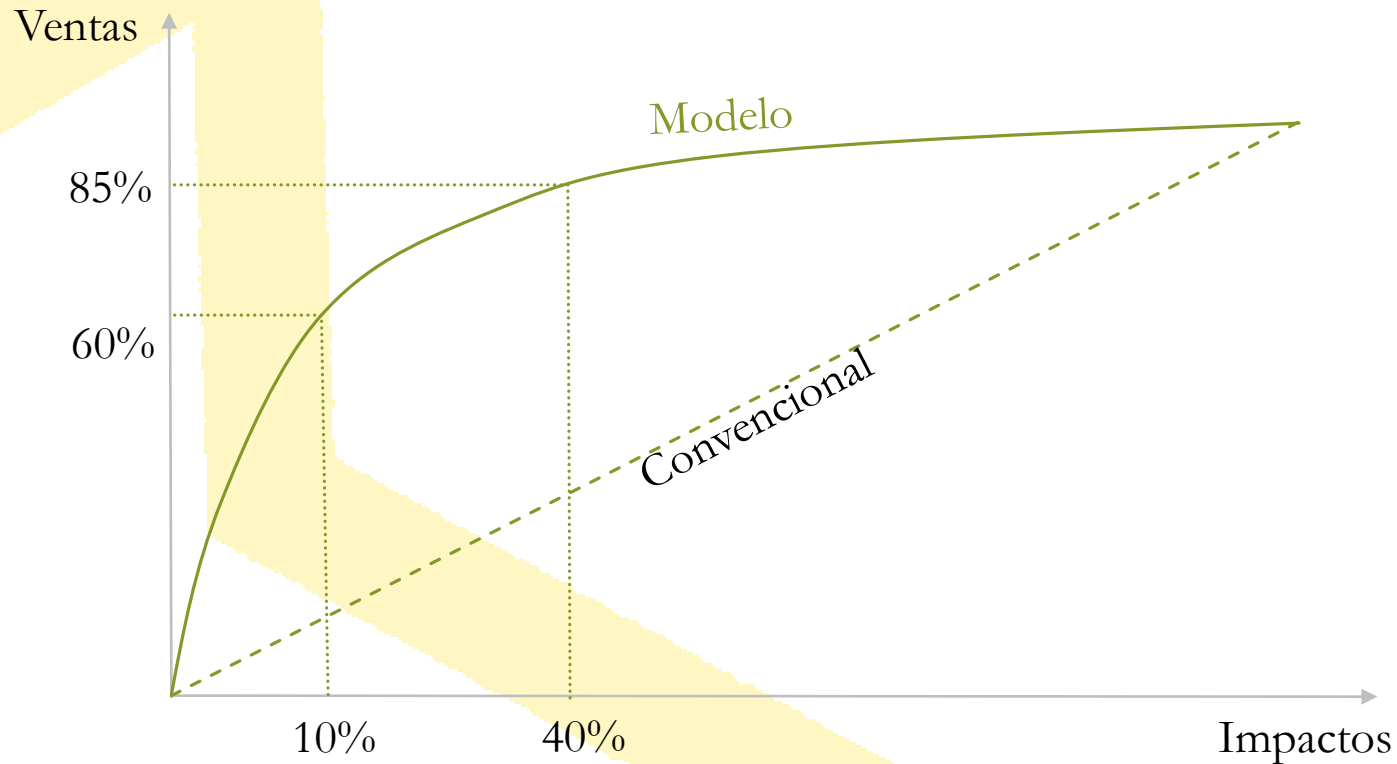
---

# Presentar los resultados

- Responder siempre al problema definido en la primera fase.
- Los resultados se presentan a profesionales de diferentes perfiles, por lo que se debe orientar a usuarios de negocio/directivos.

# Presentar los resultados

## LIFT CHART



## Agenda

1- Clasificación vs Regresión

---

2- Árboles de decisión

---

3- Bosques

---

4- K-vecinos

---

5- Ensembles

---

6- Fases de un proyecto

---

**7- Ejercicios guiados**

## Agenda

7.1- Ejercicio1\_Arboles\_Decision\_Clasificacion

---

7.2- Ejercicio2\_AirlineDelay\_Clasificación

---

7.3- Bosques

---

7.4- Ejercicio4\_HousePricesCaseStudy\_Regresion

---



## Ejercicio1\_Arboles\_Decision\_Clasificacion

### **Guión a seguir:**

- Carga de datos
- Análisis exploratorio
- Preprocesado del dataset
- Modelización
- Análisis de los resultados

## Ejercicio1\_Arboles\_Decision\_Clasificacion

### Carga de datos:

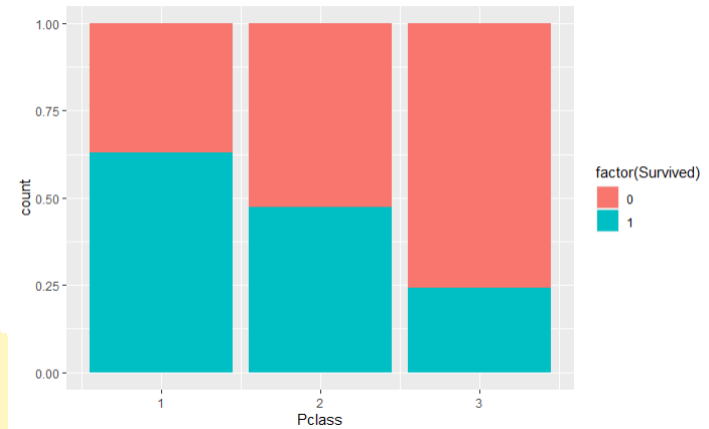
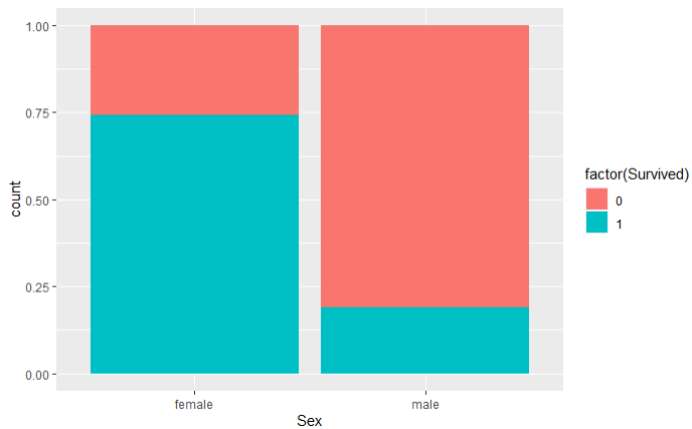
```
```{r titanic}
data_path <- 'titanic.csv'
titanic <- read.csv(data_path, header = T, stringsAsFactors = F, encoding = 'utf-8')
```
```

## Ejercicio1\_Arboles\_Decision\_Clasificacion

### Análisis exploratorio:

```
{r}  
dim(titanic)  
str(titanic)  
summary(titanic)
```

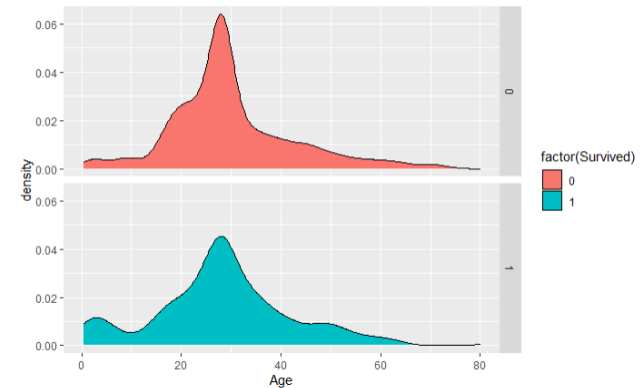
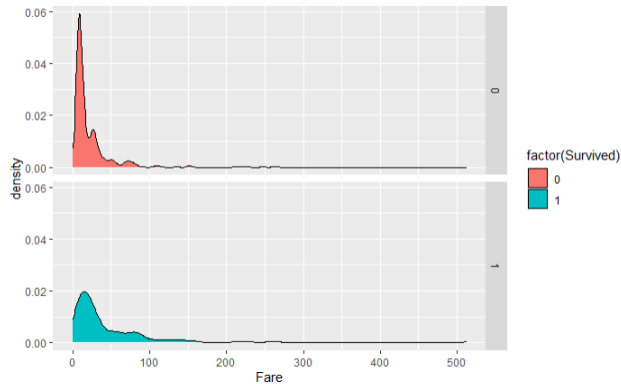
#### Algunos plots:



## Ejercicio1\_Arboles\_Decision\_Clasificacion

### Análisis exploratorio:

Algunos plots:



## Ejercicio1\_Arboles\_Decision\_Clasificacion

### **Preprocesado del dato:**

```
```{r}  
titanic$Survived <- as.factor(titanic$Survived)  
```
```

## Ejercicio1\_Arboles\_Decision\_Clasificacion

### Modelización:

```
```{r}
library(caret)
set.seed(23)
split <- 0.7
trainIndex <- createDataPartition(titanic$Survived, p=split, list=FALSE)
head(trainIndex)
```
```

```
```{r}
titanic_training <- titanic[trainIndex,]
titanic_test <- titanic[-trainIndex,]
```
```

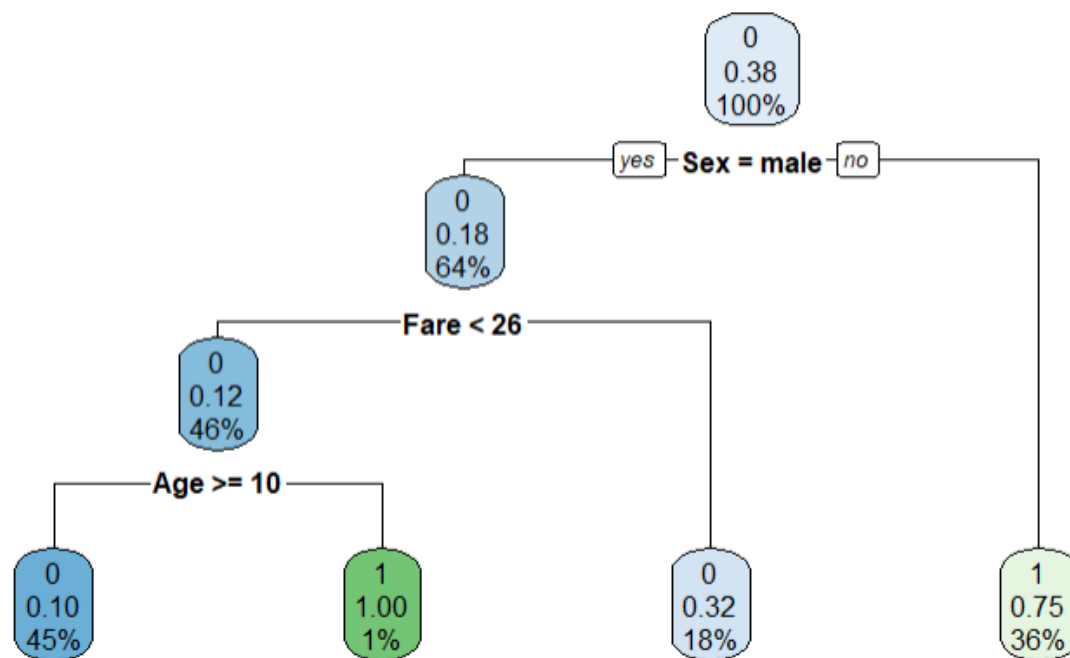
```
```{r}
library(rpart)

Primer_Arbol_Decision <- rpart(formula = Survived ~ Sex + Age, data = titanic_training)

Primer_Arbol_Decision
```
```

## Ejercicio1\_Arboles\_Decision\_Clasificacion

### Análisis de los resultados:



## Ejercicio1\_Arboles\_Decision\_Clasificacion

### Análisis de los resultados:

#### Confusion Matrix and Statistics

|            | Reference |    |
|------------|-----------|----|
| Prediction | 0         | 1  |
| 0          | 138       | 32 |
| 1          | 26        | 70 |

Accuracy : 0.782

95% CI : (0.7274, 0.8301)

No Information Rate : 0.6165

P-value [Acc > NIR] : 5.932e-09

Kappa : 0.5337

Mcnemar's Test P-Value : 0.5115

Sensitivity : 0.8415

Specificity : 0.6863

Pos Pred Value : 0.8118

Neg Pred value : 0.7292

Prevalence : 0.6165

Detection Rate : 0.5188

Detection Prevalence : 0.6391

Balanced Accuracy : 0.7639

'Positive' Class : 0



## Agenda

7.1- Ejercicio1\_Arboles\_Decision\_Clasificación

---

**7.2- Ejercicio2\_AirlineDelay\_Clasificación**

---

7.3- Bosques

---

7.4- Ejercicio4\_HousePricesCaseStudy\_Regresion

---

## Ejercicio2\_AirlineDelay\_Clasificación

### **Guión a seguir:**

- Carga de datos
- Análisis exploratorio
- Preprocesado del dataset
- Modelización
- Análisis de los resultados

## Ejercicio2\_AirlineDelay\_Clasificación

### Carga de datos:

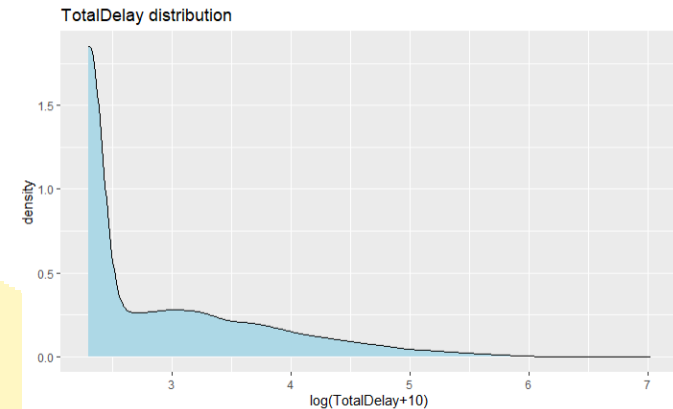
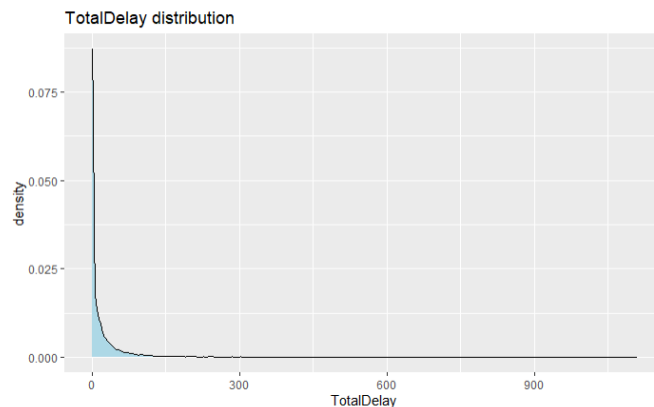
```
```{r}  
Airlines <- read.csv("AirlineDelay.csv")  
```
```

## Ejercicio2\_AirlineDelay\_Clasificación

### Análisis exploratorio:

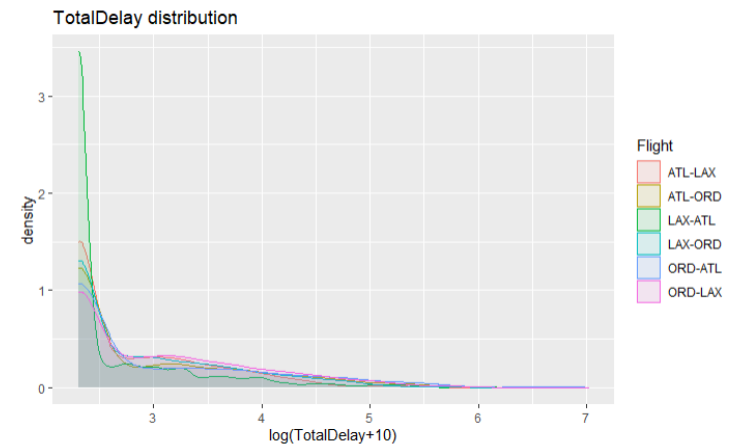
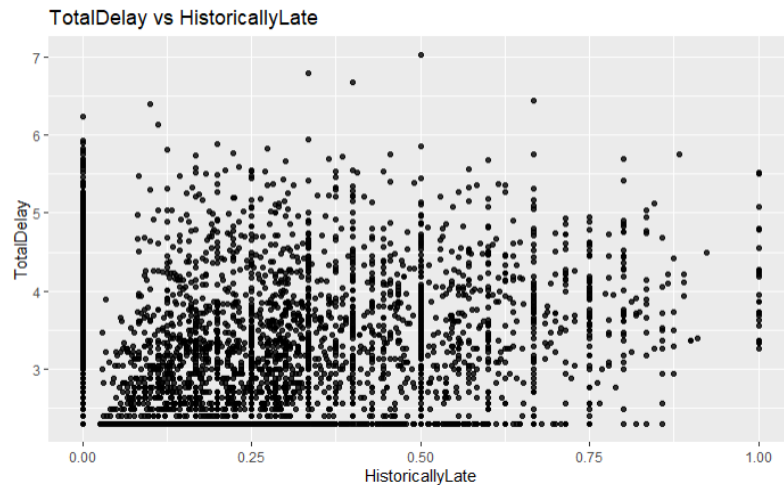
```
{r}  
names(AirlinesTrain)  
dim(AirlinesTrain)  
summary(AirlinesTrain)  
}
```

#### Algunos plots:



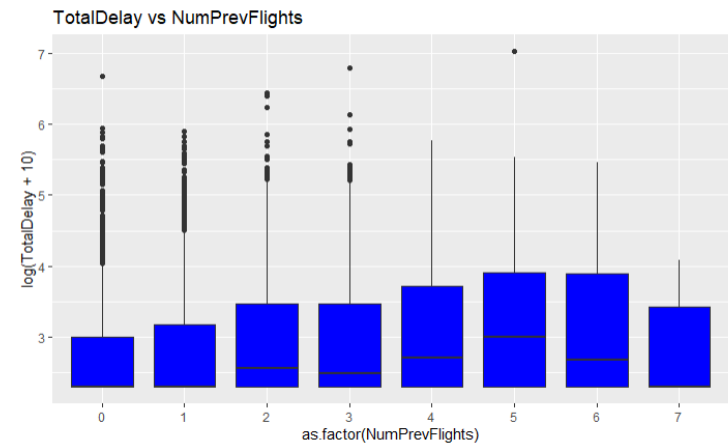
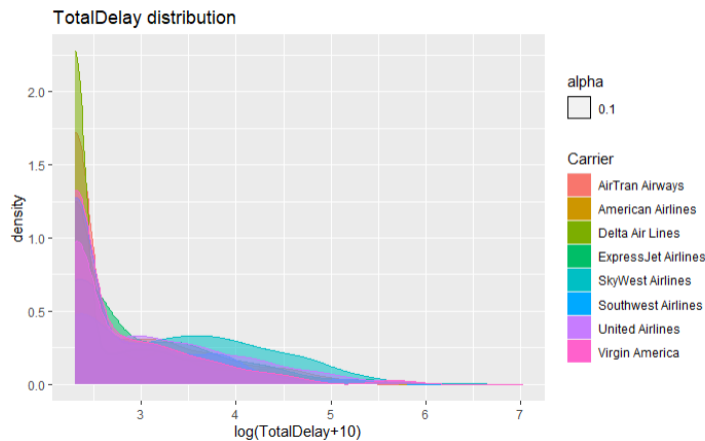
## Ejercicio2\_AirlineDelay\_Clasificación

### Análisis exploratorio:



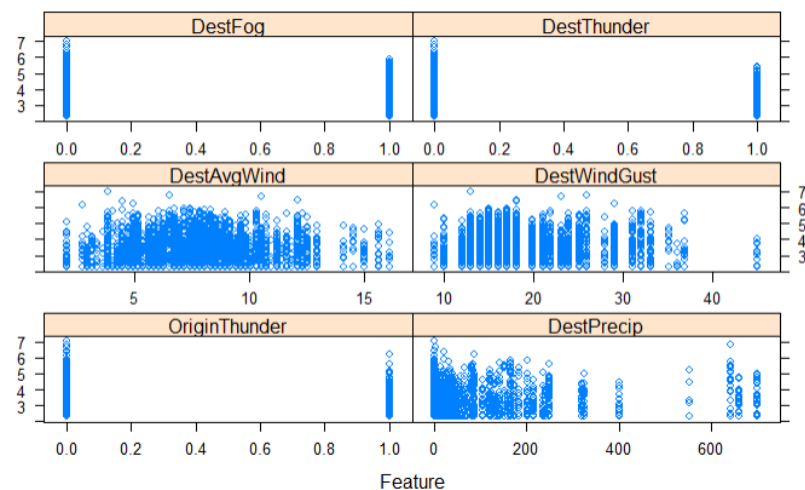
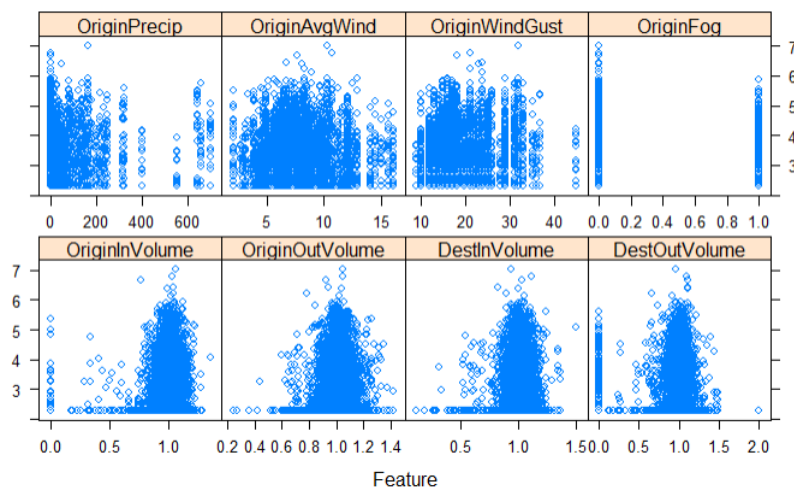
## Ejercicio2\_AirlineDelay\_Clasificación

### Análisis exploratorio:



## Ejercicio2\_AirlineDelay\_Clasificación

### Análisis exploratorio:



## Ejercicio2\_AirlineDelay\_Clasificación

### **Preprocesado del dato:**

Eliminar la variable TotalDelay, ¿Porqué?



## Ejercicio2\_AirlineDelay\_Clasificación

### Modelización:

```
```{r}
ctrl <- trainControl(method = "repeatedcv",
                     repeats = 4,
                     number = 5)
```
```

Crea un tercer modelo aplicando cross validation y el modelo Random Forest

```
```{r}
rfFit <- train(DelayClass ~ ., method = "rf",
              data = AirlinesTrain,
              preProcess = c("center", "scale"),
              trControl = ctrl)
```
```

```
```{r}
print(rfFit)
```
```

## Ejercicio2\_AirlineDelay\_Clasificación

### Análisis de los resultados:

| Prediction  | Reference   |             |          |
|-------------|-------------|-------------|----------|
|             | Major Delay | Minor Delay | No Delay |
| Major Delay | 24          | 27          | 9        |
| Minor Delay | 59          | 83          | 55       |
| No Delay    | 222         | 514         | 882      |

#### Overall statistics

Accuracy : 0.5275  
 95% CI : (0.5046, 0.5503)  
 No Information Rate : 0.5045  
 P-Value [Acc > NIR] : 0.02479

Kappa : 0.099

McNemar's Test P-Value : < 2e-16

#### Statistics by Class:

|                      | Class: Major Delay | Class: Minor Delay | Class: No Delay |
|----------------------|--------------------|--------------------|-----------------|
| Sensitivity          | 0.07869            | 0.13301            | 0.9323          |
| Specificity          | 0.97707            | 0.90887            | 0.2078          |
| Pos Pred Value       | 0.40000            | 0.42132            | 0.5451          |
| Neg Pred Value       | 0.84518            | 0.67759            | 0.7510          |
| Prevalence           | 0.16267            | 0.33280            | 0.5045          |
| Detection Rate       | 0.01280            | 0.04427            | 0.4704          |
| Detection Prevalence | 0.03200            | 0.10507            | 0.8629          |
| Balanced Accuracy    | 0.52788            | 0.52094            | 0.5700          |

## Agenda

7.1- Ejercicio1\_Arboles\_Decision\_Clasificacion

---

7.2- Ejercicio2\_AirlineDelay\_Clasificación

---

**7.3- Ejercicio3\_AirlineDelay\_Regresion**

---

7.4- Ejercicio4\_HousePricesCaseStudy\_Regresion

---

## Ejercicio3\_AirlineDelay\_Regresion

### **Guión a seguir:**

- Carga de datos
- Análisis exploratorio
- Preprocesado del dataset
- Modelización
- Análisis de los resultados

## Ejercicio3\_AirlineDelay\_Regresion

### Carga de datos:

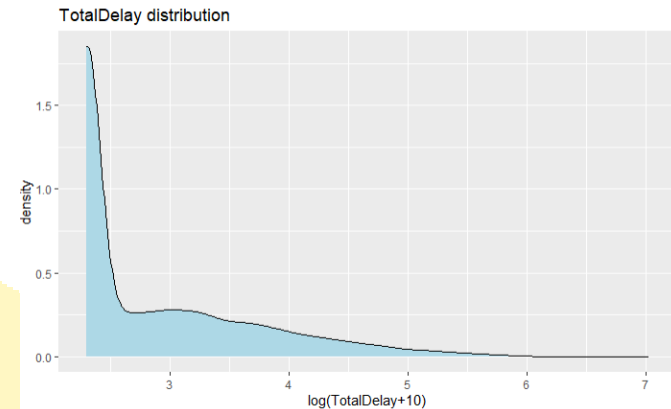
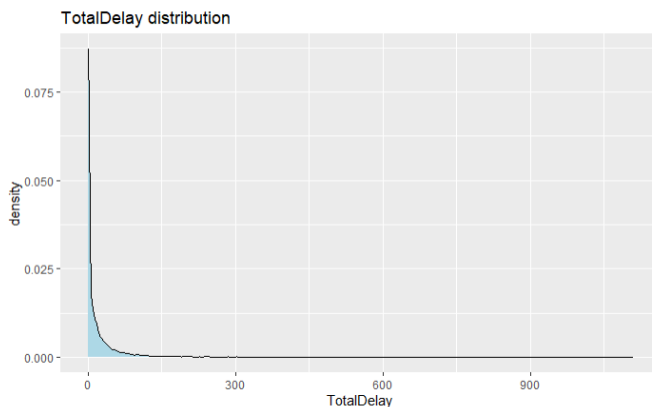
```
```{r}  
Airlines <- read.csv("AirlineDelay.csv")  
```
```

## Ejercicio3\_AirlineDelay\_Regresion

### Análisis exploratorio:

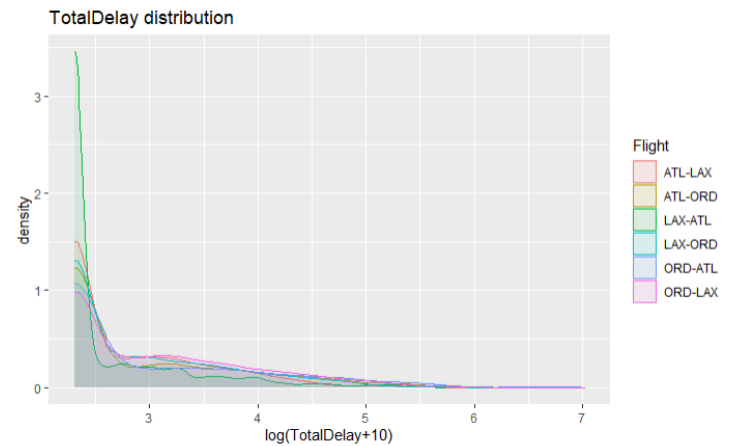
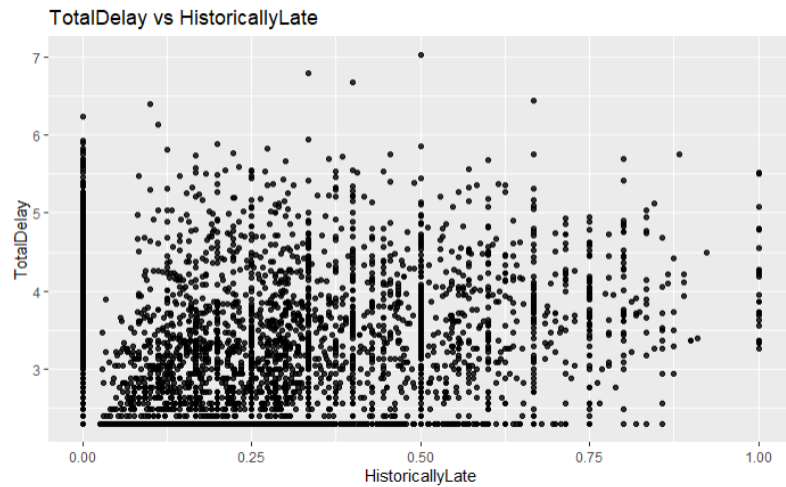
```
{r}  
names(AirlinesTrain)  
dim(AirlinesTrain)  
summary(AirlinesTrain)  
}
```

#### Algunos plots:



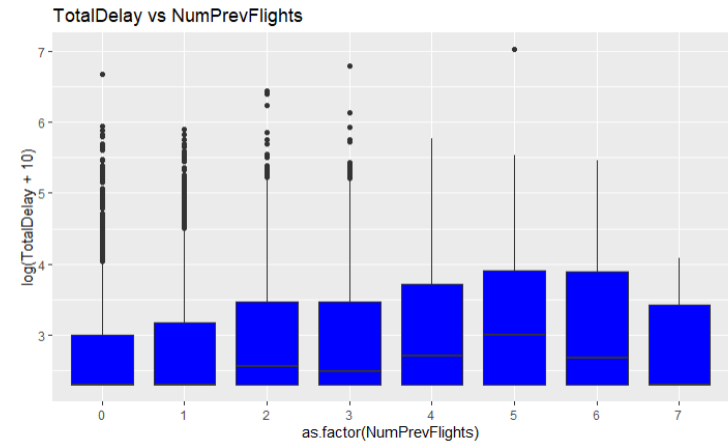
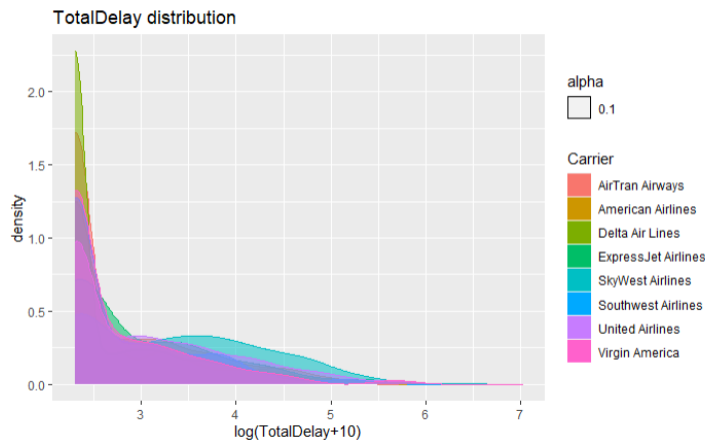
## Ejercicio3\_AirlineDelay\_Regresion

### Análisis exploratorio:



## Ejercicio3\_AirlineDelay\_Regresion

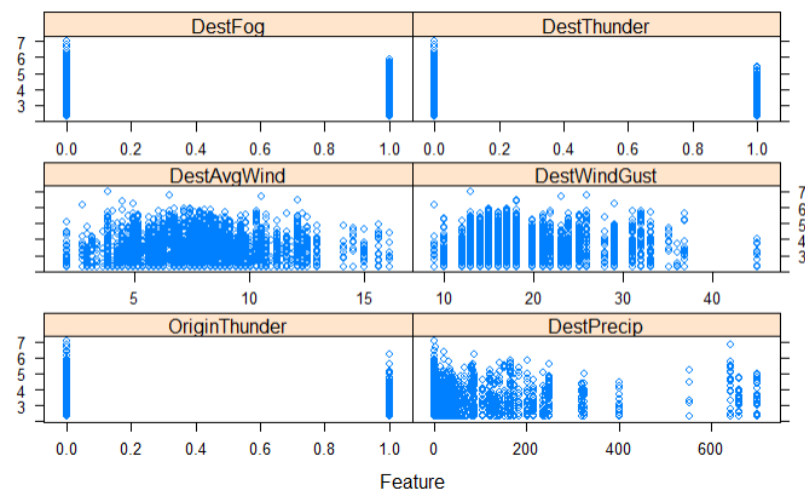
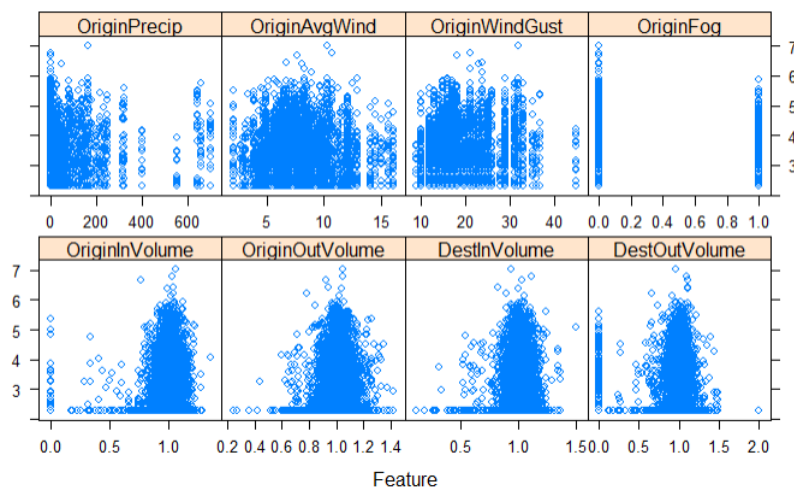
### Análisis exploratorio:





## Ejercicio3\_AirlineDelay\_Regresion

### Análisis exploratorio:



## Ejercicio3\_AirlineDelay\_Regresion

### **Preprocesado del dato:**

Eliminar la variable TotalDelay, ¿Porqué?

## Ejercicio3\_AirlineDelay\_Regresion

### Modelización:

```
```{r}
ctrl <- trainControl(method = "repeatedcv",
                     repeats = 4,
                     number = 5)
...

```{r}
Tercer_Modelo_rf_tune <- train(ModelF, data = AirlinesTrain,
                              method = "rf",
                              trControl = ctrl,
                              ntree = 20,
                              tuneGrid = expand.grid(mtry = c(1,2,3,4,5)),
                              verbose = FALSE)
...
```
```

## Ejercicio3\_AirlineDelay\_Regresion

### Modelización:

Crea un cuarto modelo, con Random Forest, y prueba a tunear el modelo para encontrar la mejor precisión

```
```{r}
control <- trainControl(method="repeatedcv", number=10, repeats=3)
seed <- 7
metric <- "Accuracy"
set.seed(seed)
mtry <- sqrt(ncol(x))
tuneGrid <- expand.grid(.mtry=mtry)
rf_default <- train(TotalDelay~., data=dataset, method="rf", metric=metric, tuneGrid=tuneGrid, trControl=control)
print(rf_default)
```
```

Prueba a tunear un modelo con Grid search

```
```{r}
control <- trainControl(method="repeatedcv", number=10, repeats=3, search="grid")
set.seed(seed)
tuneGrid <- expand.grid(.mtry=c(1:15))
rf_gridsearch <- train(TotalDelay~., data=dataset, method="rf", metric=metric, tuneGrid=tuneGrid, trControl=control)
print(rf_gridsearch)
plot(rf_gridsearch)
```
```

Prueba a tunear un modelo manualmente

```
```{r}
control <- trainControl(method="repeatedcv", number=10, repeats=3, search="grid")
tuneGrid <- expand.grid(.mtry=c(sqrt(ncol(x))))
modellist <- list()
for (ntree in c(1000, 1500, 2000, 2500)) {
  set.seed(seed)
  fit <- train(TotalDelay~., data=dataset, method="rf", metric=metric, tuneGrid=tuneGrid, trControl=control,
    ntree=ntree)
  key <- toString(ntree)
  modellist[[key]] <- fit
}

results <- resamples(modellist)
summary(results)
dotplot(results)
```
```

## Ejercicio3\_AirlineDelay\_Regresion

### Análisis de los resultados:

```
Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:
    Min       1Q   Median       3Q      Max
-1.9595 -0.4869 -0.1966  0.3625  3.8508

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.884005   0.008347  345.503 < 2e-16 ***
`FlightATL-ORD` -0.001459   0.011696   -0.125  0.900746
`FlightLAX-ATL` -0.002286   0.011760   -0.194  0.845869
`FlightLAX-ORD` -0.034491   0.083787   -0.412  0.680608
`FlightORD-ATL`  0.006913   0.012234    0.565  0.572055
`FlightORD-LAX`  0.012857   0.082553    0.156  0.876238
`CarrierAmerican Airlines` 0.023675   0.081074    0.292  0.770276
`CarrierDelta Air Lines` -0.012178   0.028462   -0.428  0.668773
`CarrierExpressJet Airlines` 0.018265   0.010215    1.788  0.073813 .
`CarrierSkywest Airlines`  0.027672   0.013230    2.092  0.036505 *
`CarrierSouthwest Airlines` -0.006475   0.015680   -0.413  0.679659
`CarrierUnited Airlines`  0.082814   0.085408    0.970  0.332261
`CarrierVirgin America`  0.013487   0.039952    0.338  0.735701
MonthJuly      -0.059115   0.010109   -5.848  5.20e-09 ***
MonthJune      -0.007934   0.010318   -0.769  0.441970
DayOfWeekMonday  0.007080   0.011122    0.637  0.524393
DayOfWeekSaturday -0.028242   0.011287   -2.502  0.012368 *
DayOfWeekSunday -0.041037   0.011534   -3.558  0.000376 ***
DayOfWeekThursday  0.025938   0.011330    2.289  0.022083 *
DayOfWeekTuesday -0.013827   0.011572   -1.195  0.232182
DayOfWeekWednesday 0.021225   0.011274    1.883  0.059772 .
NumPrevFlights  0.045167   0.011655    3.875  0.000107 ***
PrevFlightGap    0.004722   0.011460    0.412  0.680300
HistoricallyLate  0.276224   0.011318   24.405 < 2e-16 ***
InsufficientHistory 0.170624   0.011699   14.584 < 2e-16 ***
OriginInVolume   0.014493   0.008723    1.662  0.096650 .
OriginOutVolume  0.024832   0.008846    2.807  0.005011 **
DestInVolume     0.022424   0.008931    2.511  0.012071 *
DestOutVolume    0.012733   0.009111    1.398  0.162296
OriginPrecip     0.057812   0.009482    6.097  1.13e-09 ***
OriginAvgWind    -0.036038   0.011327   -3.182  0.001471 **
OriginWindGust   0.097375   0.011162    8.724 < 2e-16 ***
OriginFog        0.011183   0.008971    1.247  0.212581
OriginThunder    0.013291   0.008710    1.526  0.127088
DestPrecip       0.072504   0.009515    7.620  2.86e-14 ***
DestAvgWind     -0.028084   0.011383   -2.467  0.013639 *
               5.134  2.91e-07 ***
               2.318  0.020503 **
               3.337  0.000852 ***
```

Residual standard error: 0.7232 on 7467 degrees of freedom  
Multiple R-squared: 0.1842, Adjusted R-squared: 0.18  
F-statistic: 44.35 on 38 and 7467 DF, p-value: < 2.2e-16

0.1 ' ' 1

## Agenda

7.1- Ejercicio1\_Arboles\_Decision\_Clasificacion

---

7.2- Ejercicio2\_AirlineDelay\_Clasificación

---

7.3- Ejercicio3\_AirlineDelay\_Regresion

---

**7.4- Ejercicio4\_HousePricesCaseStudy\_Regresion**

---

## Ejercicio4\_HousePricesCaseStudy\_Regresion

### **Guión a seguir:**

- Carga de datos
- Análisis exploratorio
- Preprocesado del dataset
- Modelización
- Análisis de los resultados

## Ejercicio4\_HousePricesCaseStudy\_Regresion

### Carga de datos:

```
```{r titanic}  
data(Sacramento)  
```
```

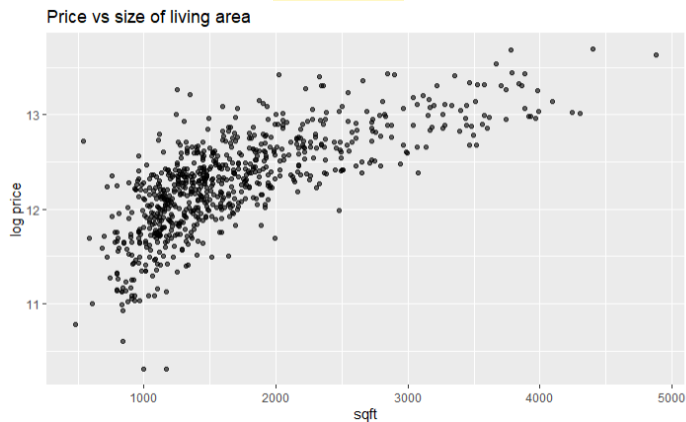


## Ejercicio4\_HousePricesCaseStudy\_Regresion

### Análisis exploratorio:

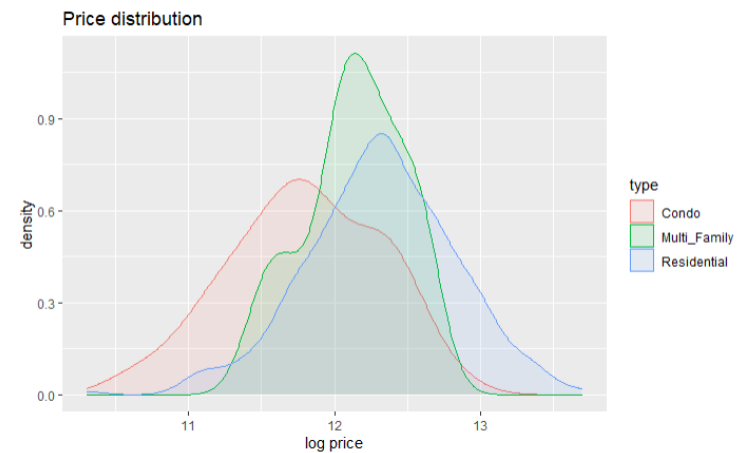
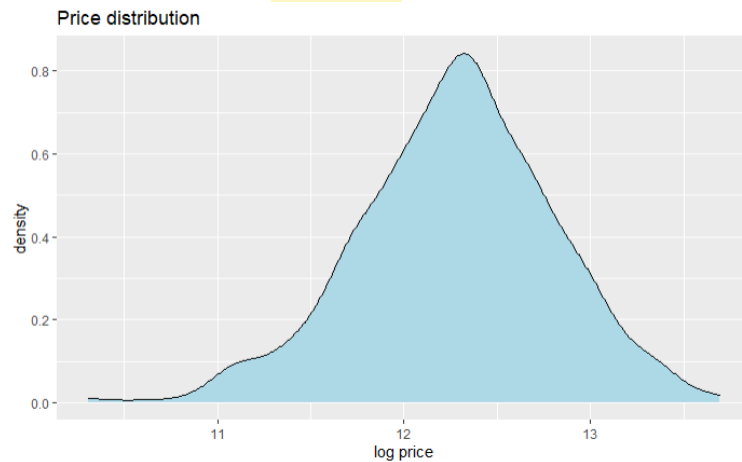
```
```{r}  
names(Sacramento)  
dim(Sacramento)  
str(Sacramento)  
summary(Sacramento)  
```
```

### Algunos plots:



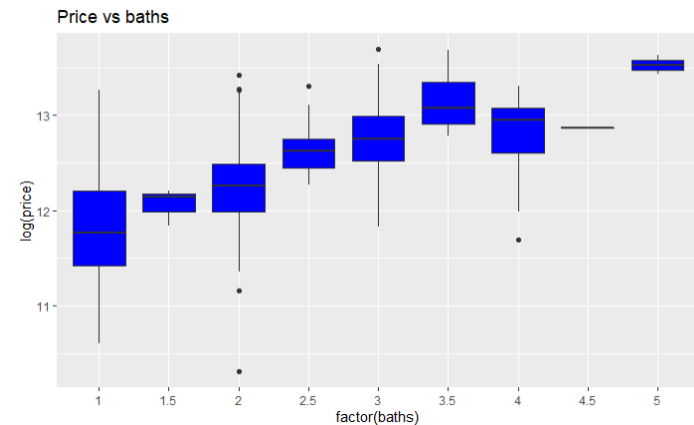
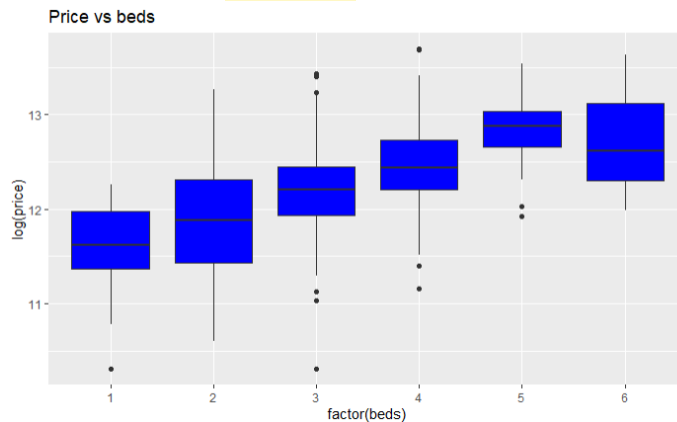
## Ejercicio4\_HousePricesCaseStudy\_Regresion

### Análisis exploratorio:



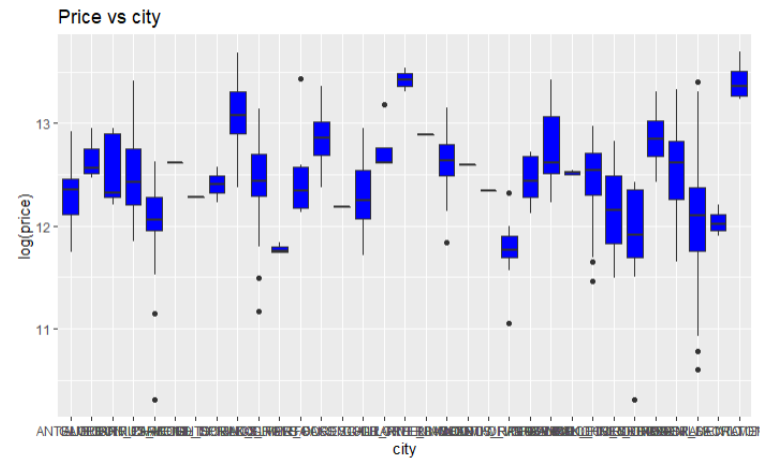
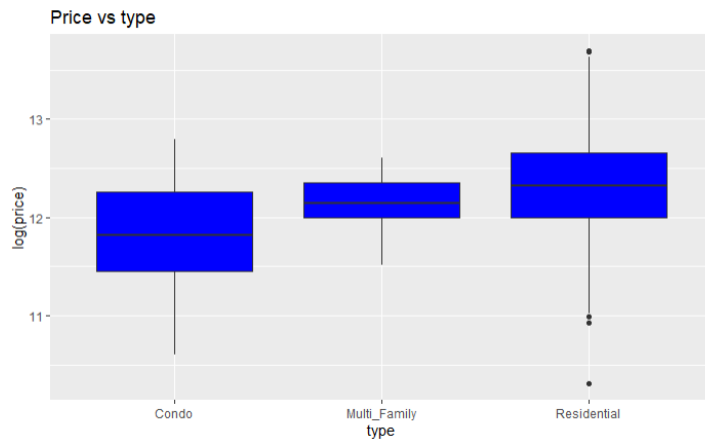
## Ejercicio4\_HousePricesCaseStudy\_Regresion

### Análisis exploratorio:



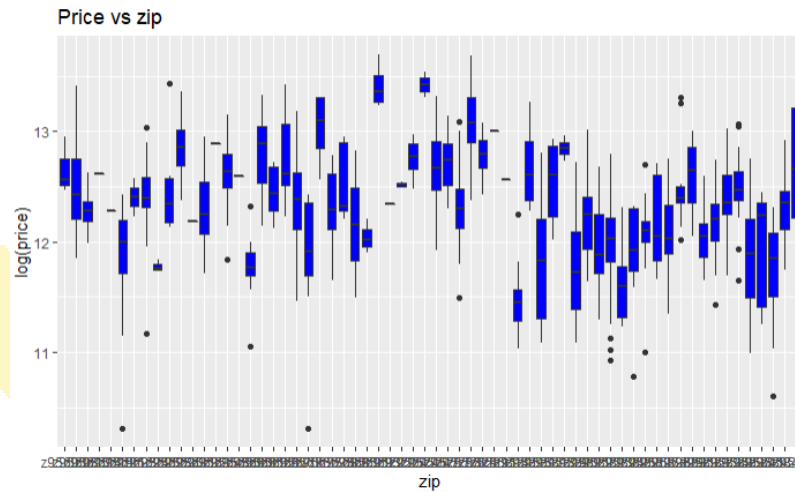
## Ejercicio4\_HousePricesCaseStudy\_Regresion

### Análisis exploratorio:



## Ejercicio4\_HousePricesCaseStudy\_Regresion

### Análisis exploratorio:



## Ejercicio4\_HousePricesCaseStudy\_Regresion

### Preprocesado del dato:

Los missing values son especialmente peligrosos si se encuentran en el dataset de test si tiene algunos NAs en una variables, eliminalos  
si por el contrario, tienes muchos NAs en una variable, elimina la variable  
Algunas veces, conviene reemplazar estos NAs por la media o la mediana  
Y por supuesto, utiliza el sentido común para aplicar estas normas

En este caso, vamos a realizar ingeniería de variables:

```
```{r}
pairs(cbind(training$sqft,training$beds/training$baths))

training$bedsperbath = training$beds/training$baths
testing$bedsperbath = testing$beds/testing$baths
```
```

## Ejercicio4\_HousePricesCaseStudy\_Regresion

### Modelización:

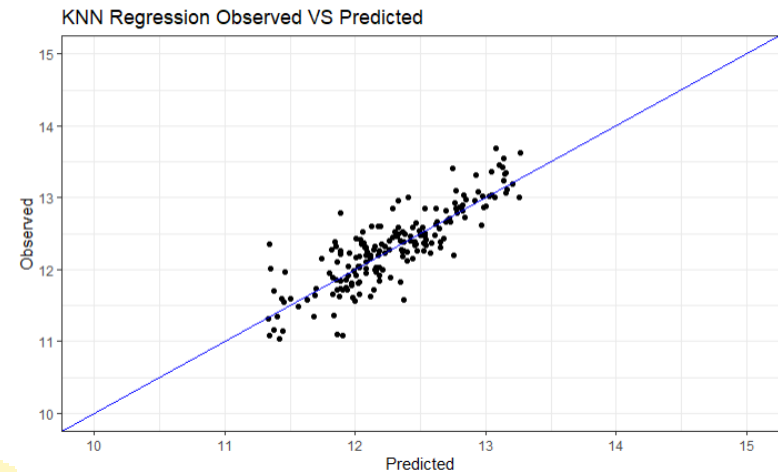
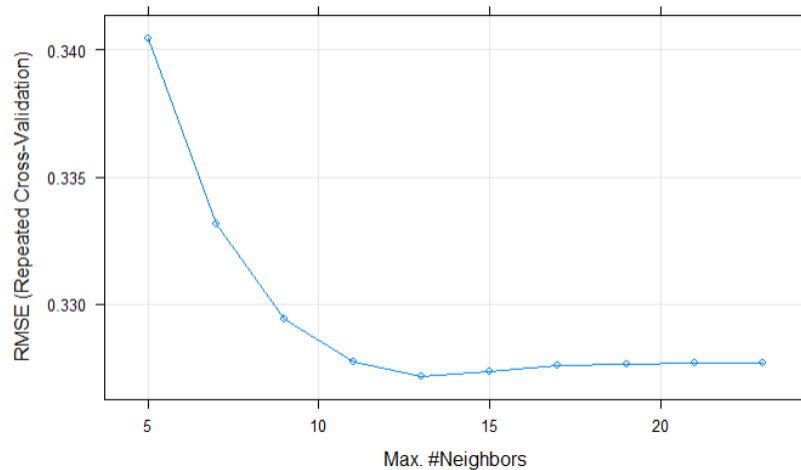
```
```{r}
knn_tune <- train(ModelF, data = training,
  method = "knnn",
  preProc=c('scale','center'),
  tuneLength = 10,
  trControl = ctrl)
```
```

```
```{r}
rf_tune <- train(ModelF, data = training,
  method = "rf",
  preProc=c('scale','center'),
  trControl = ctrl,
  #ntree = 20,
  tuneGrid = expand.grid(mtry = c(10,20,30)),
  verbose = FALSE)
```
```

## Ejercicio4\_HousePricesCaseStudy\_Regresion

### Análisis de los resultados:

K-vecinos:





## Ejercicio4\_HousePricesCaseStudy\_Regresion

### Análisis de los resultados:

Random Forest:

