

**{desafío}**  
**latam\_**

# Máquinas de Soporte Vectorial \_



# Motivación

# ¿Qué son?

Las máquinas de soporte vectorial son un marco analítico que entrega soluciones a problemas de regresión y clasificación.

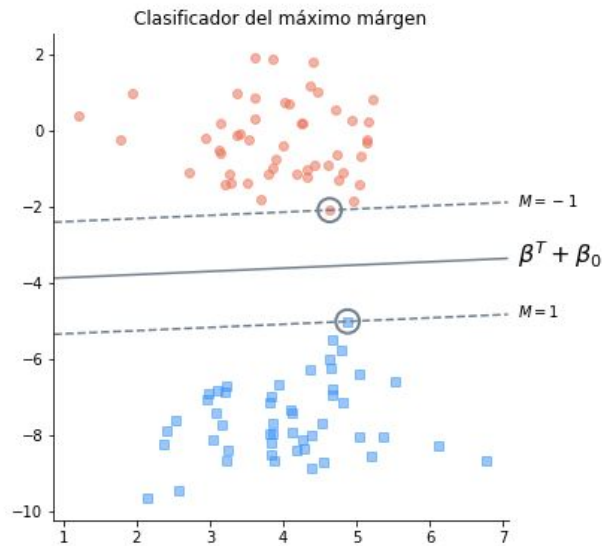
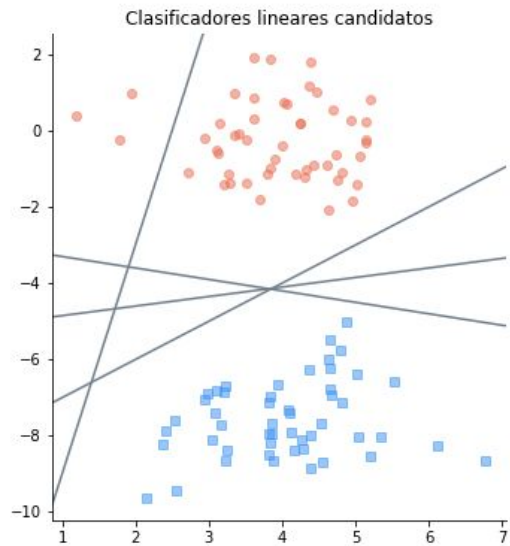
Presentan varias virtudes en comparación a otros modelos:

- Presentan una implementación rápida.
- Buenos resultados predictivos.
- Permite enfrentar problemas de no linealidad.
- Se beneficia de las matrices dispersas.

# Clasificadores de Máximo Margen

# Casos separables

- SVM es un modelo discriminativo: **generamos una función candidata y posteriormente la implementamos en un conjunto de datos.**
- El problema es que en un conjunto de datos, puede existir más de una función candidata que separe las clases de un conjunto.
- Este problema se conoce como **no identificabilidad** de la función candidata.
- SVM implementa el principio del máximo margen para encontrar un clasificador lineal óptimo.
- Busca aumentar el margen existente en la distancia entre las dos nubes de datos.



## Obtención del plano

$$\left\{ x : f(x) = \mathbf{X}^T \beta + \beta_0 = 0 \right\}$$

- El plano satisface una función lineal.

# Obtención del plano

$$\left\{ x : f(x) = \mathbf{X}^T \beta + \beta_0 = 0 \right\}$$

- El plano satisface una función lineal.
- La principal diferencia con métodos como MCO o EMV, es que éste se optimiza mediante la maximización del margen.



# Obtención del plano

$$G(x) = \text{sign} \left[ \mathbf{X}^T \beta + \beta_0 \right]$$

- El plano satisface una función lineal.
- La principal diferencia con métodos como MCO o EMV, es que éste se optimiza mediante la maximización del margen.
- Si tenemos un problema binario con clases -1 y 1, el clasificador responderá si es que una observación se posiciona sobre o bajo el plano.

# Obtención del plano

$$\operatorname{argmax}_{\beta, \beta_0, ||\beta||=1} M$$

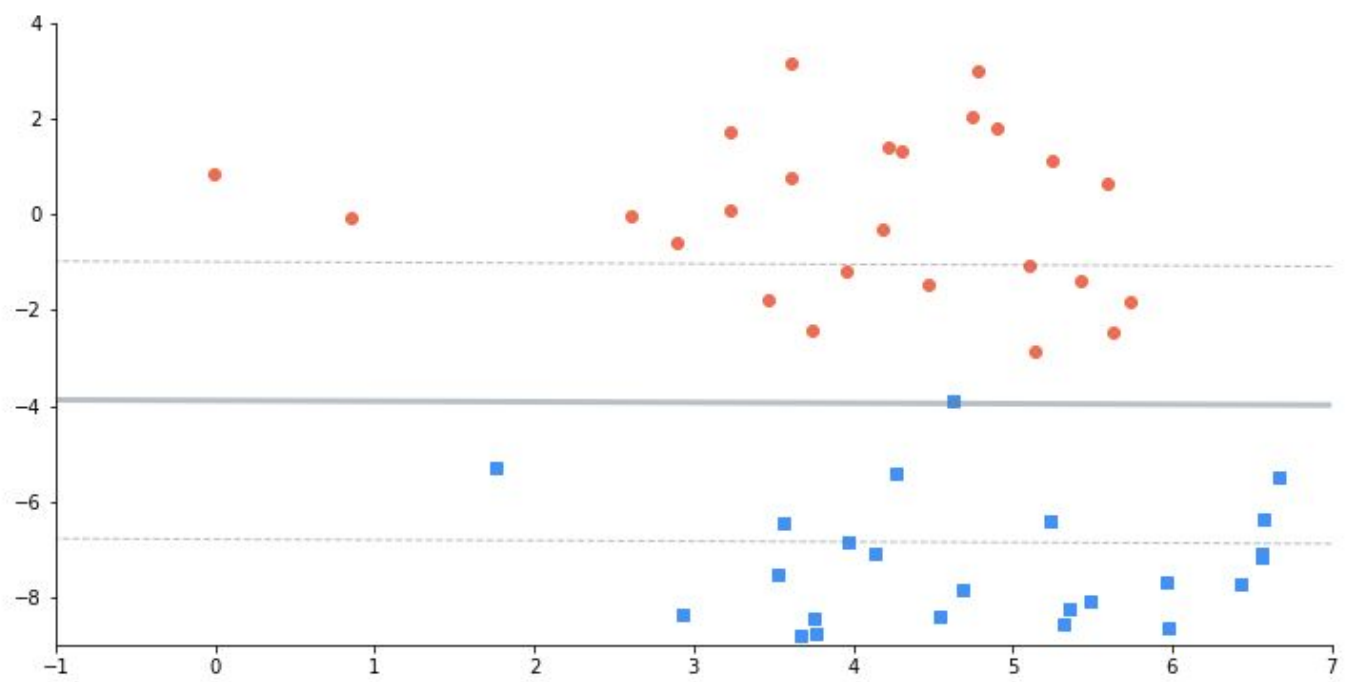
$$\text{Subjeto a: } y_i(\mathbf{X}^T \beta + \beta_0) \geq M \quad \forall i \in N$$

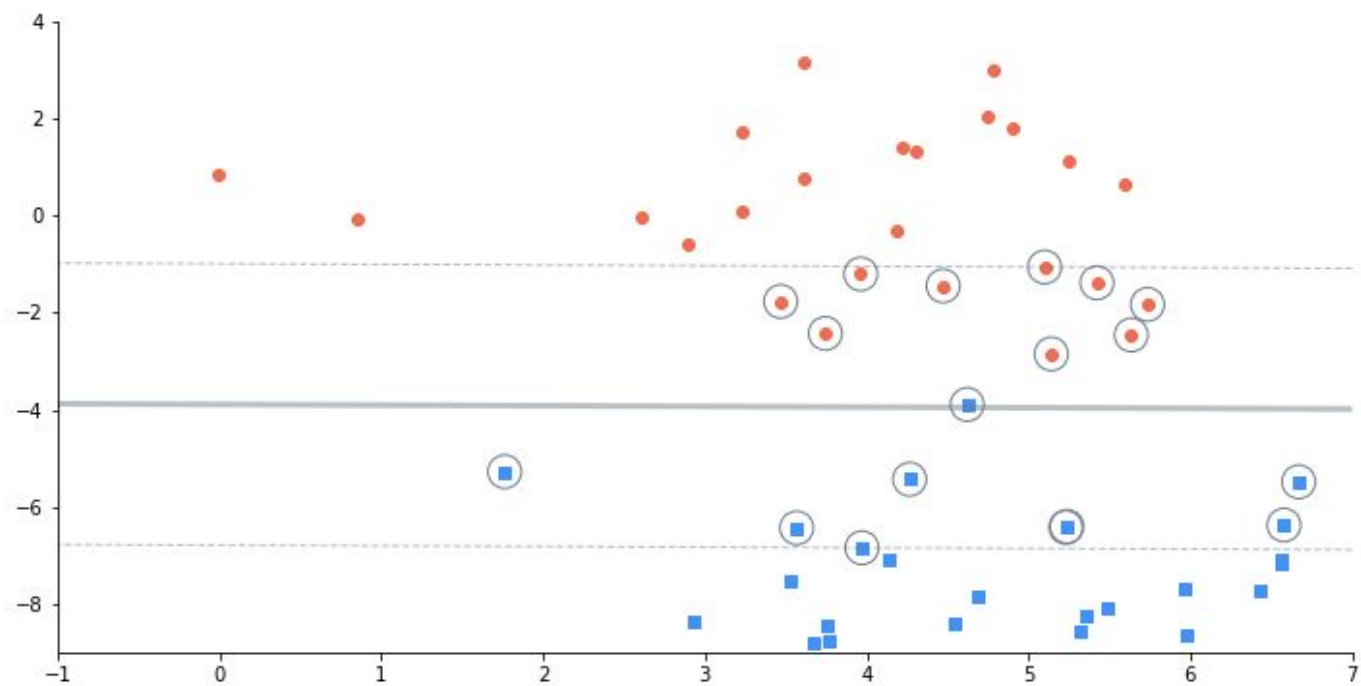
- El plano satisface una función lineal.
- La principal diferencia con métodos como MCO o EMV, es que éste se optimiza mediante la maximización del margen.
- Si tenemos un problema binario con clases -1 y 1, el clasificador responderá si es que una observación se posiciona sobre o bajo el plano.

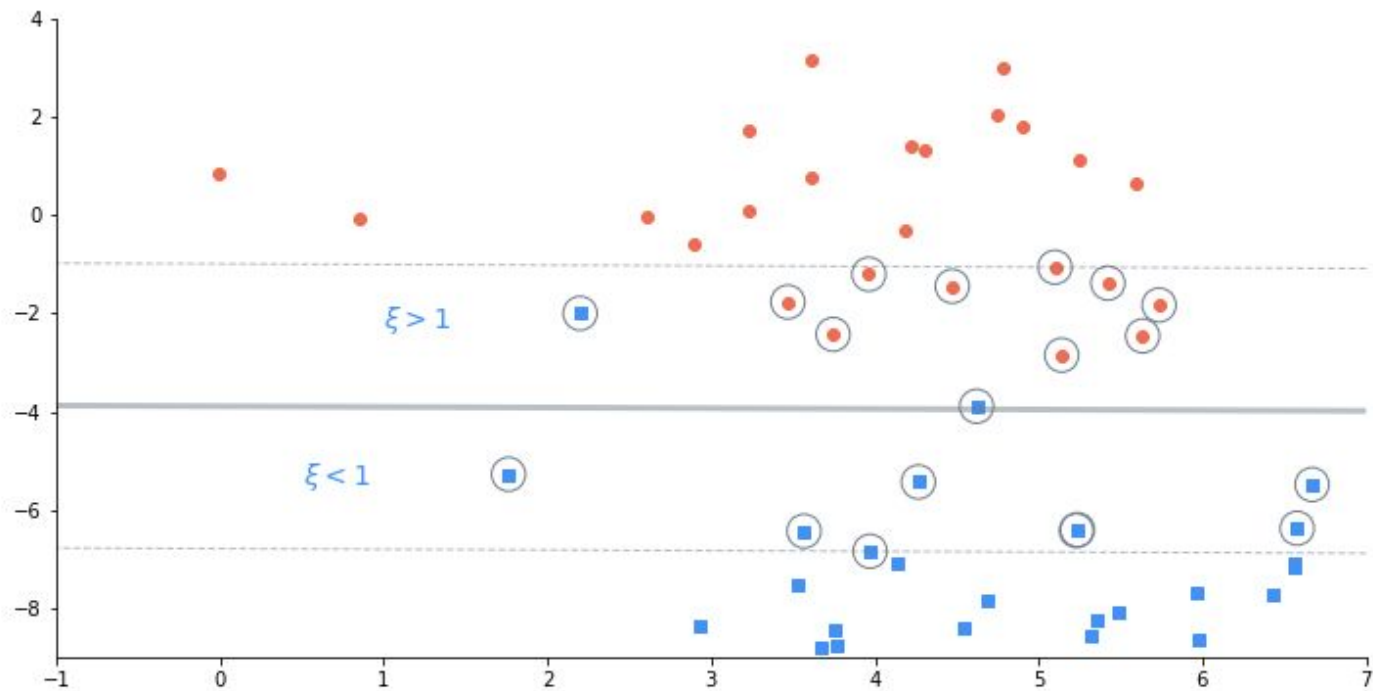
# Máquinas de Soporte Vectorial

# Casos No Separables

- Posterior a la definición del plano de división, el segundo elemento es considerar la tolerancia a observaciones predichas incorrectamente.
- Éstas se materializan en variables slacks:
  - $\xi > 1$  : Observación se posiciona sobre el margen de su clase. Situación esperable.
  - $\xi < 1$  : Observación cruza el clasificador lineal y se posiciona en el margen de la clase contraria. Situación a minimizar.







# Estandarización de atributos

- SVM se apoya en vectores para dimensionar el margen existente en una tarea de clasificación.
- El problema es que los vectores dependerán del rango de los atributos.
- Si tenemos rangos distintos en nuestra matriz de atributos en entrenamiento, puede que el modelo no sirva para la predicción de nuevas observaciones.
- Por lo general podemos implementar la estandarización/normalización de éstos.



# Pipelining

## Solución sin Pipelines

```
X_std=StandardScaler().fit_transform()  
  
X_dim_red=PCA(n_components=3).fit_transform(X_std)  
  
model=LinearRegression().fit(X_dim_red, y_train)
```

## Solución con Pipelines

```
pipeline_model = Pipeline(  
    [('scale', StandardScaler()),  
     ('pca', PCA(n_components=3)),  
     ('model', LinearRegression())]  
)  
  
pipeline_model.fit(X_train, y_train)
```

# Kernelización

# ¿Qué es un kernel?

- De manera adicional a flexibilizar los slacks en nuestro margen, podemos reexpresar nuestros datos en un nuevo espacio donde sí sean linealmente separables.
- **Esto se logra mediante un kernel:** permiten a las funciones operar en altas dimensiones mediante el cálculo del producto interno de cada par de datos en el espacio de atributos.

# ¿Cómo funciona un kernel?

$$\kappa(x, x^T) = \exp(-\gamma \|x - x^T\|^2)$$

$$\mathbf{X} = \begin{bmatrix} (x_1, x_1) & \cdots & (x_1, x_n) \\ \vdots & \ddots & \vdots \\ (x_n, x_1) & \cdots & (x_m, x_n) \end{bmatrix}$$

Implementaremos un kernel que nos permita reexpresar la similitud entre puntos.

De esta manera, explotaremos la comunalidad entre los atributos en un espacio de mayor dimensionalidad.

Tenemos una matriz de atributos a reexpresar.

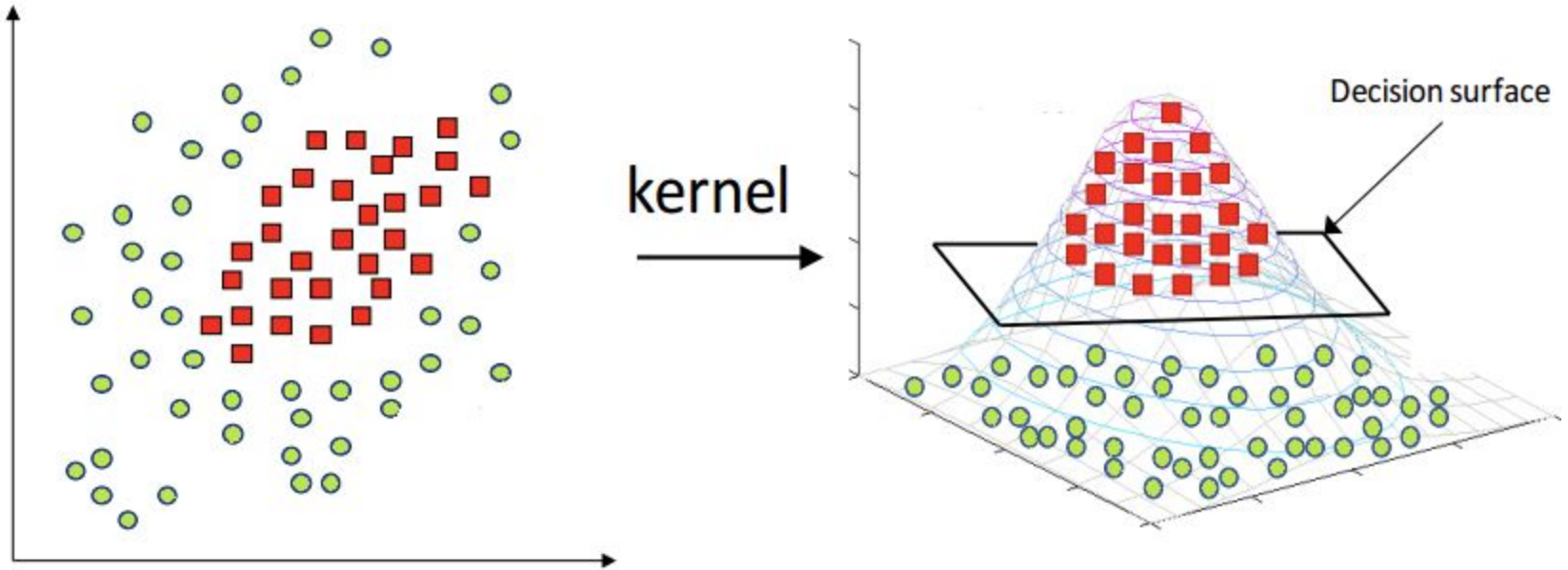
# ¿Cómo funciona un kernel?

$$\kappa(x, x^T) = \exp(-\gamma \|x - x^T\|^2)$$

$$\mathbf{X} = \begin{bmatrix} (x_1, x_1) & \cdots & (x_1, x_n) \\ \vdots & \ddots & \vdots \\ (x_n, x_1) & \cdots & (x_m, x_n) \end{bmatrix}$$

$$\mathbf{X} = \begin{bmatrix} \kappa(x_1, x_1) & \cdots & \kappa(x_1, x_n) \\ \vdots & \ddots & \vdots \\ \kappa(x_n, x_1) & \cdots & \kappa(x_m, x_n) \end{bmatrix}$$

# Visualización



# Hiperparámetros

# Hiperparámetros en SVM

**Costo:** Maneja el trueque entre la penalización de observaciones clasificadas de forma incorrectas y la estabilización de la función de decisión.

**Gamma:** Maneja la influencia de una observación específica en la función de decisión.

Ambos hiperparámetros apuntan a la misma dirección: modificar la tolerancia del plano ante los vectores de soporte y slacks.



# Búsqueda de Hiperparámetros en Grilla

	$C$			
$\gamma$	(100, 0.0001)	(10,0.0001)	(5,0.0001)	(1,0.0001)
	(100,0.001)	(10,0.001)	(5,0.001)	(1,0.001)
	(100,0.01)	(10,0.01)	(5,0.0.01)	(1,0.01)
	(100,0.1)	(10,0.1)	(5,0.1)	(1,0.1)

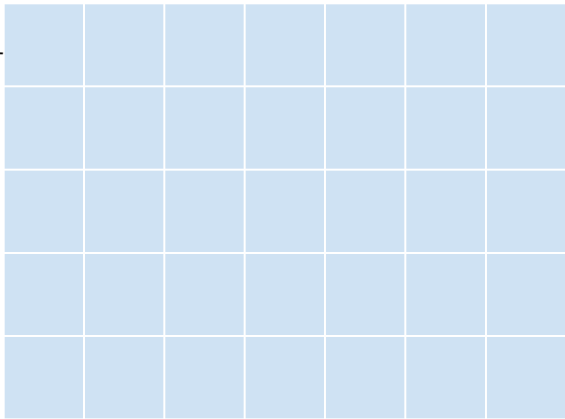
	$C$			
$\gamma$	(100, 0.0001)	(10, 0.0001)	(5, 0.0001)	(1, 0.0001)
	(100, 0.001)	(10, 0.001)	(5, 0.001)	(1, 0.001)
	(100, 0.01)	(10, 0.01)	(5, 0.01)	(1, 0.01)
	(100, 0.1)	(10, 0.1)	(5, 0.1)	(1, 0.1)

$$\#Modelos = \prod_{i=1}^{Hiperparams} \xi_i \times \#CantidadCV$$

	$C$			
$\gamma$	(100, 0.0001)	(10, 0.0001)	(5, 0.0001)	(1, 0.0001)
	(100, 0.001)	(10, 0.001)	(5, 0.001)	(1, 0.001)
	(100, 0.01)	(10, 0.01)	(5, 0.01)	(1, 0.01)
	(100, 0.1)	(10, 0.1)	(5, 0.1)	(1, 0.1)

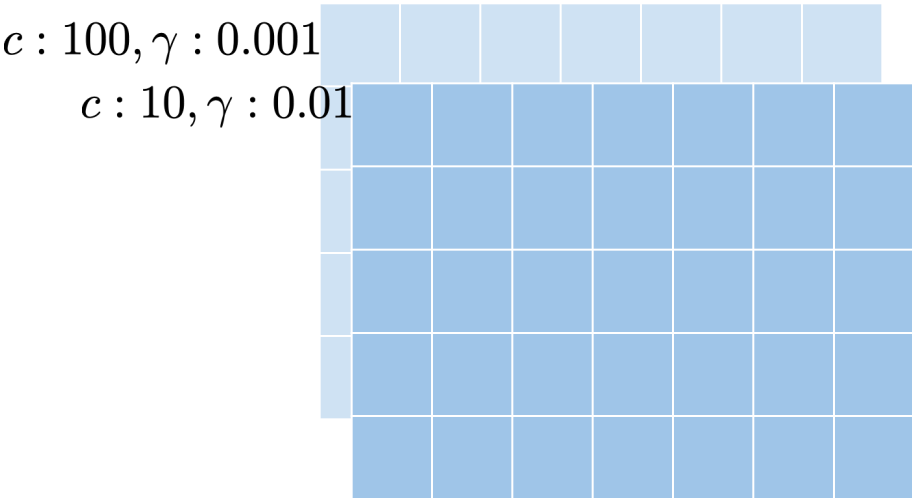
$$\# \text{Modelos} = \prod_{i=1}^{\text{Hiperparams}} \xi_i \times \# \text{CantidadCV}$$

$c : 100, \gamma : 0.001$



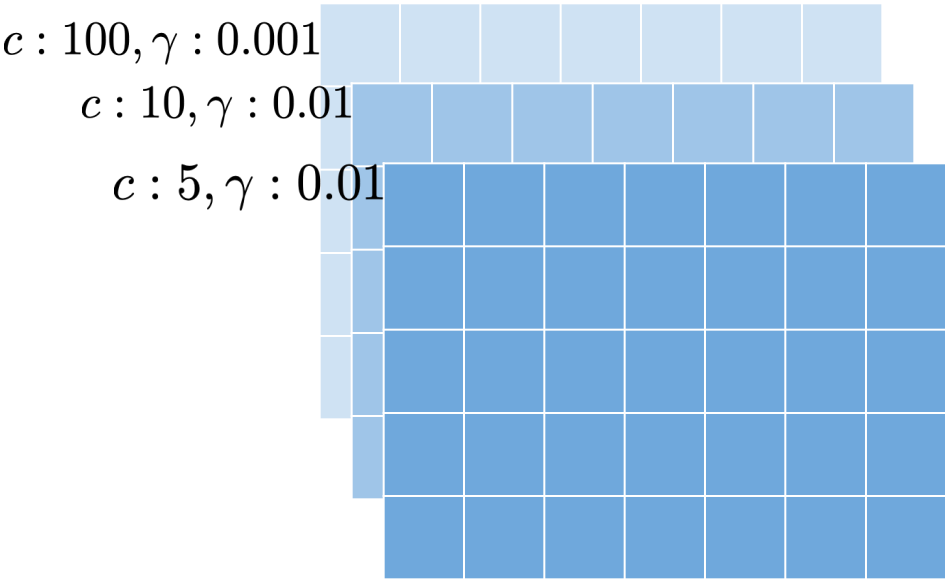
	$C$			
$\gamma$	(100, 0.0001)	(10, 0.0001)	(5, 0.0001)	(1, 0.0001)
	(100, 0.001)	(10, 0.001)	(5, 0.001)	(1, 0.001)
	(100, 0.01)	(10, 0.01)	(5, 0.01)	(1, 0.01)
	(100, 0.1)	(10, 0.1)	(5, 0.1)	(1, 0.1)

$$\# \text{Modelos} = \prod_{i=1}^{\text{Hiperparams}} \xi_i \times \# \text{CantidadCV}$$



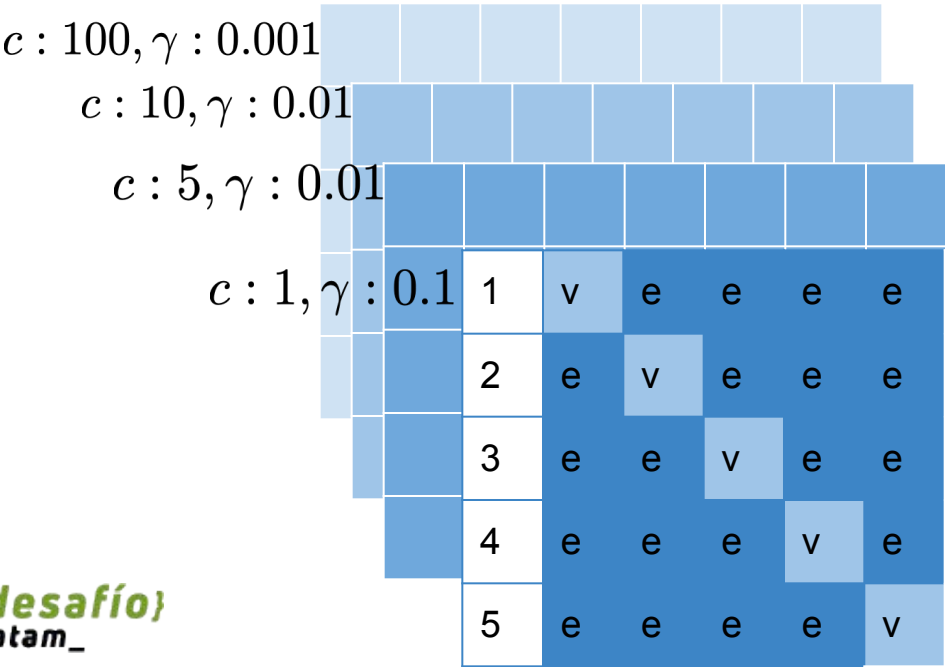
	$C$			
$\gamma$	(100, 0.0001)	(10, 0.0001)	(5, 0.0001)	(1, 0.0001)
	(100, 0.001)	(10, 0.001)	(5, 0.001)	(1, 0.001)
	(100, 0.01)	(10, 0.01)	(5, 0.01)	(1, 0.01)
	(100, 0.1)	(10, 0.1)	(5, 0.1)	(1, 0.1)

$$\# \text{Modelos} = \prod_{i=1}^{\text{Hiperparams}} \xi_i \times \# \text{CantidadCV}$$



	$C$			
$\gamma$	(100, 0.0001)	(10, 0.0001)	(5, 0.0001)	(1, 0.0001)
	(100, 0.001)	(10, 0.001)	(5, 0.001)	(1, 0.001)
	(100, 0.01)	(10, 0.01)	(5, 0.01)	(1, 0.01)
	(100, 0.1)	(10, 0.1)	(5, 0.1)	(1, 0.1)

$$\# \text{Modelos} = \prod_{i=1}^{\text{Hiperparams}} \xi_i \times \# \text{CantidadCV}$$







$C$ 
 $\gamma$ 

Hiperparams

$$\#Modelos = \prod_{i=1} \xi_i \times \#CantidadCV$$

 $c : 100, \gamma : 0.001$ 
 $c : 10, \gamma : 0.01$ 
 $c : 5, \gamma : 0.01$ 
 $c : 1, \gamma : 0.1$ 
 $\hat{\theta}_1$ 
 $\hat{\theta}_2$ 
 $\hat{\theta}_3$ 
 $\hat{\theta}_4$ 

**“Mejor modelo” a evaluar en nuevos datos.**

**{desafío}**  
latam\_

1	v	e	e	e	e
2	e	v	e	e	e
3	e	e	v	e	e
4	e	e	e	v	e
5	e	e	e	e	v



# División Trietápica

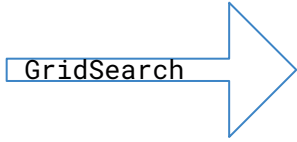
# El escenario

Conjunto  
de Datos

# El escenario

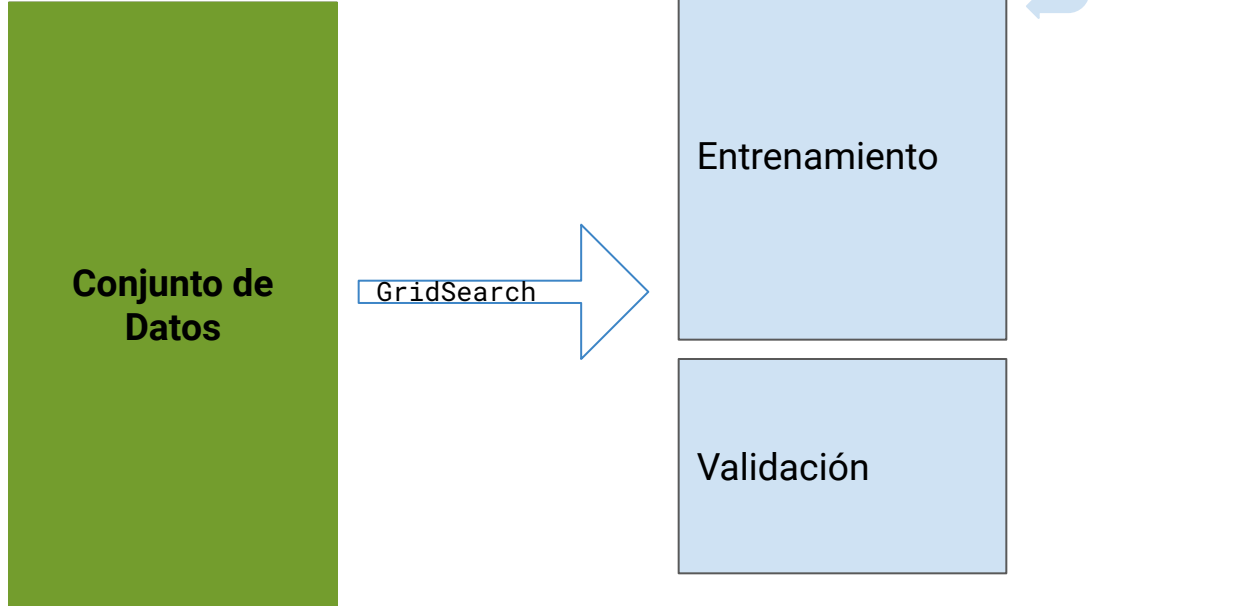
Conjunto de  
Datos

GridSearch

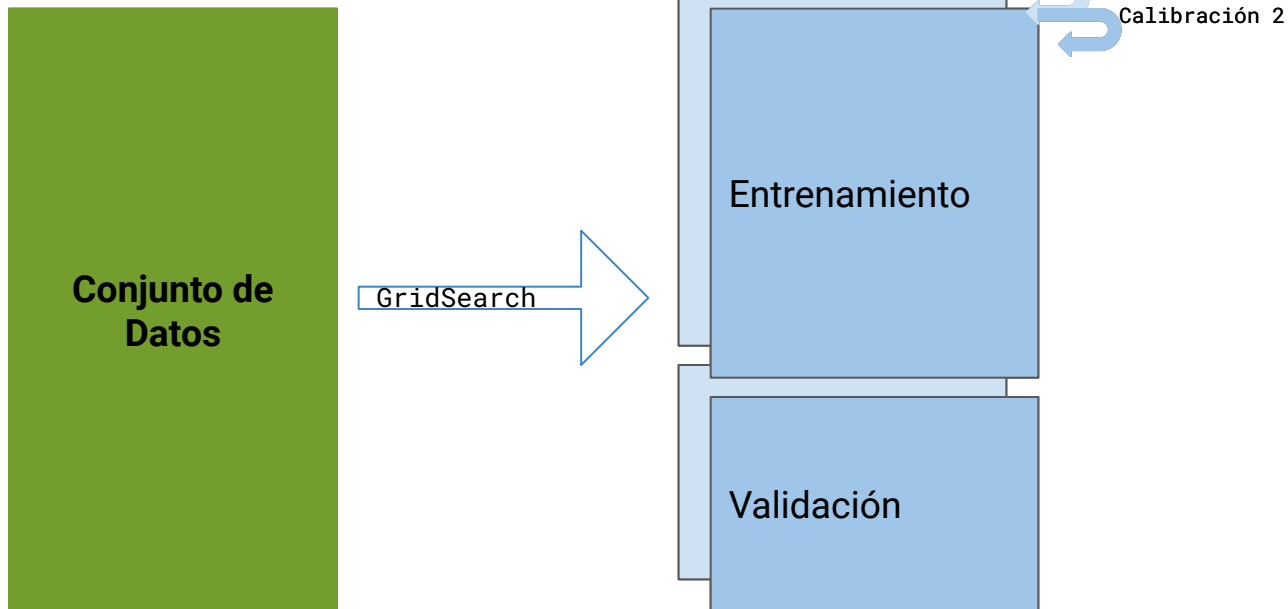


```
graph LR; A[Conjunto de Datos] -- GridSearch --> B[ ]
```

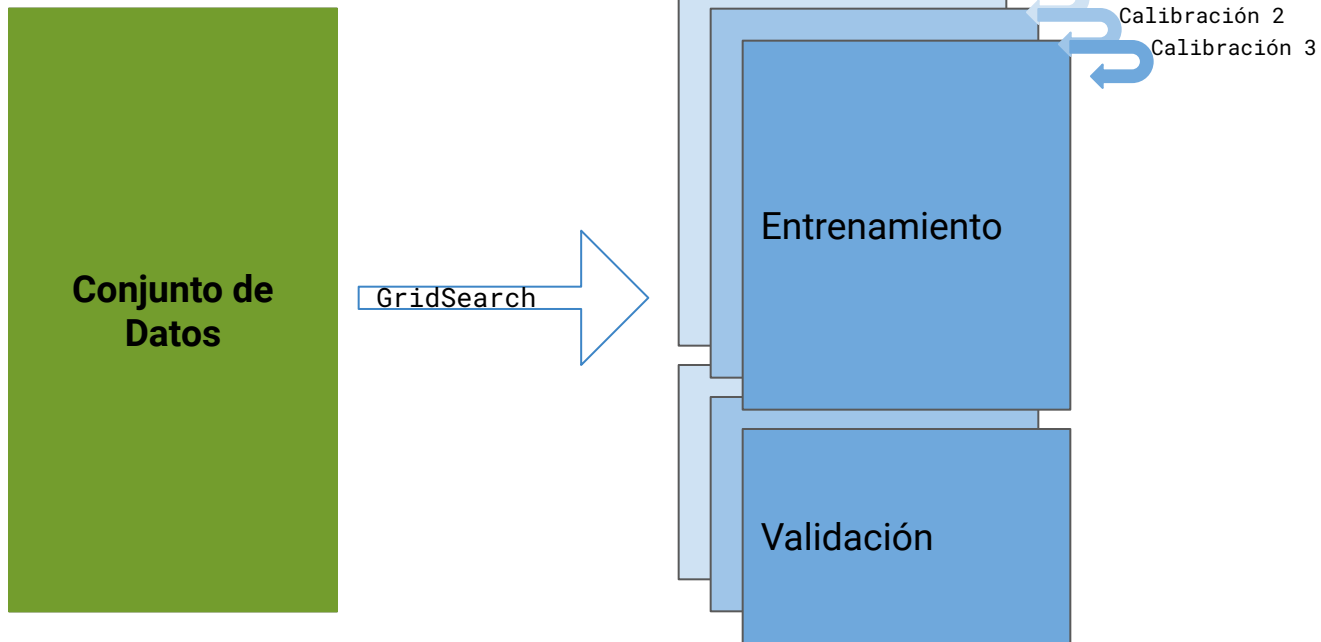
# El escenario



# El escenario

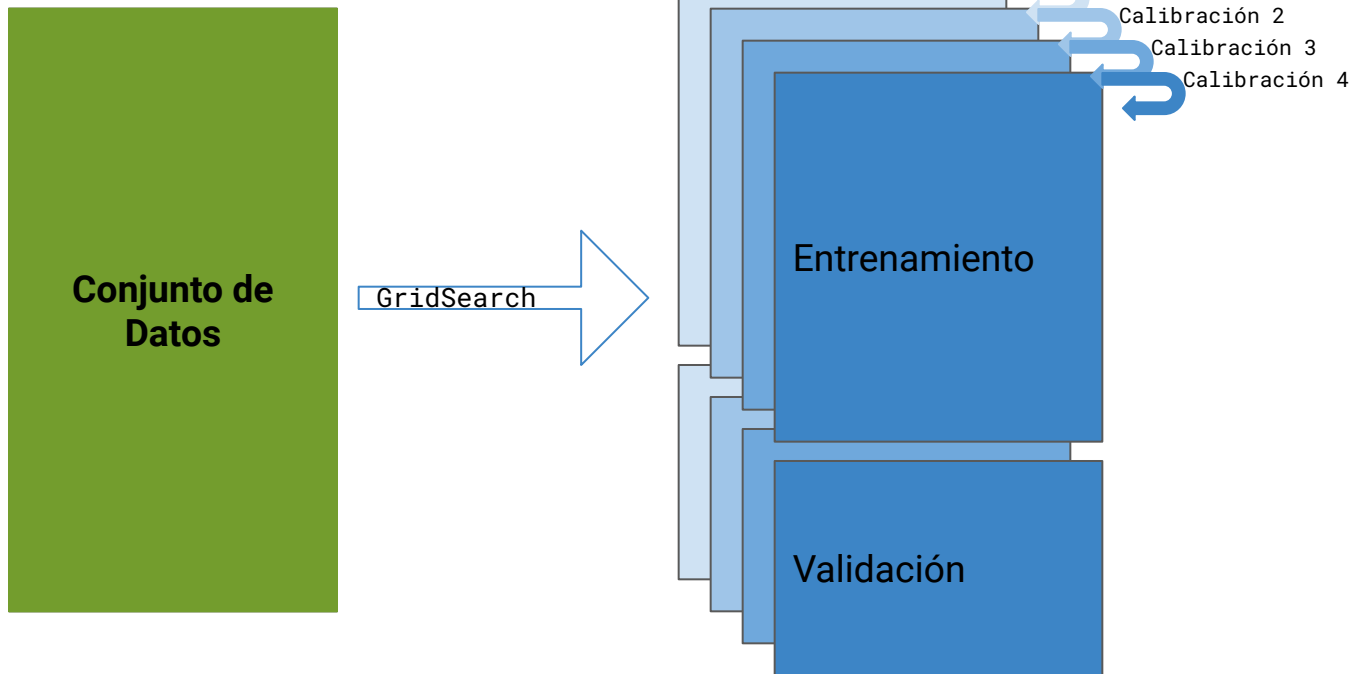


# El escenario

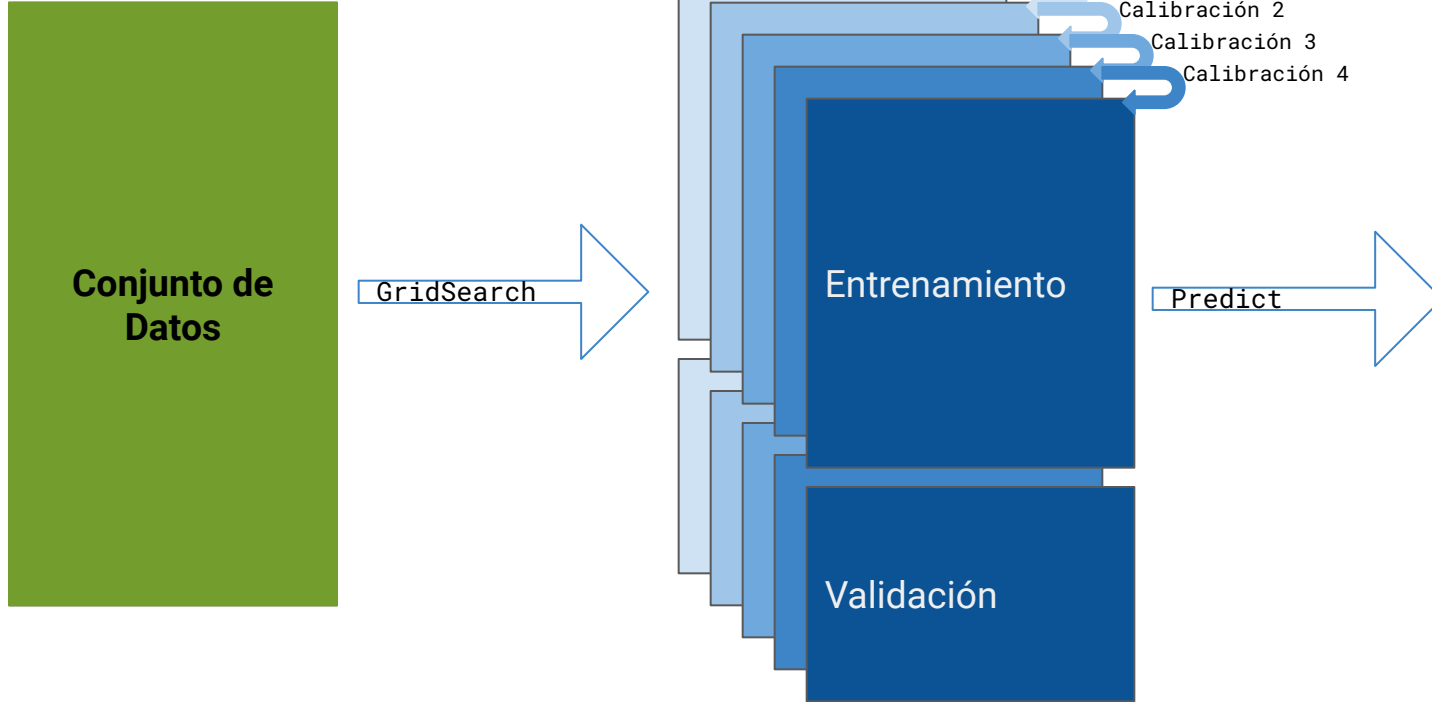




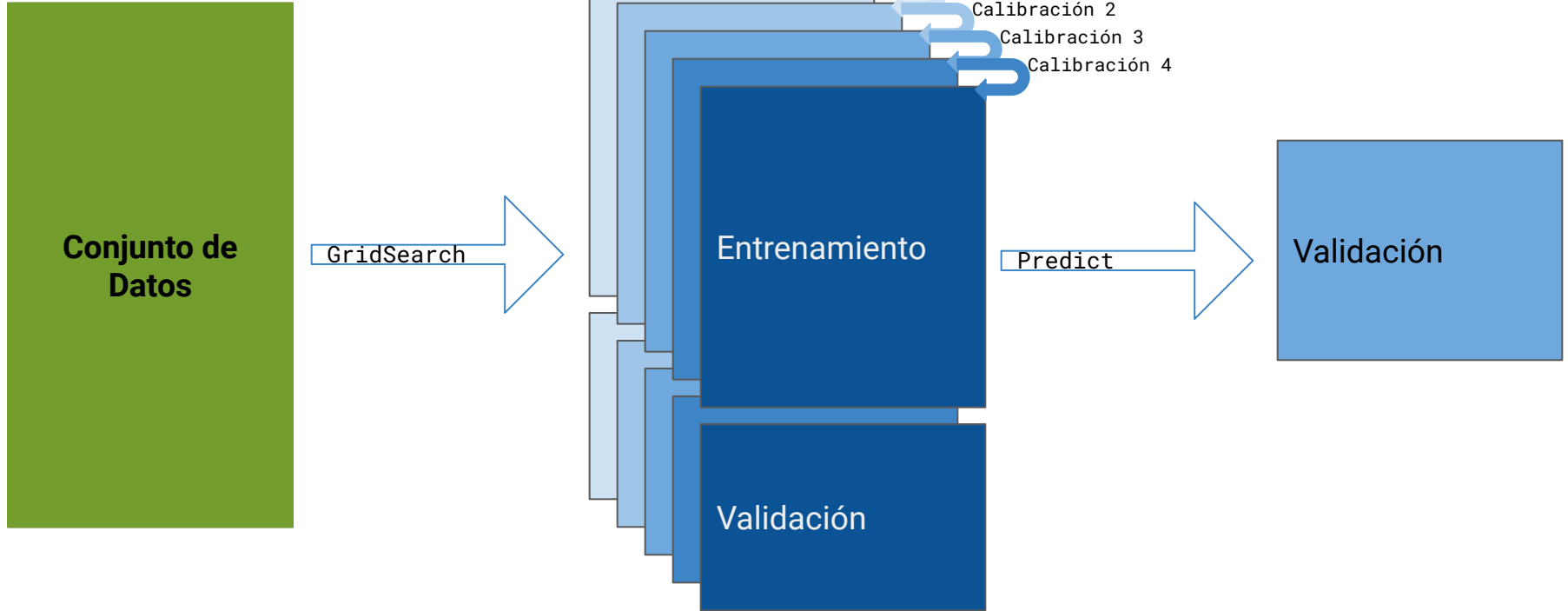
# El escenario



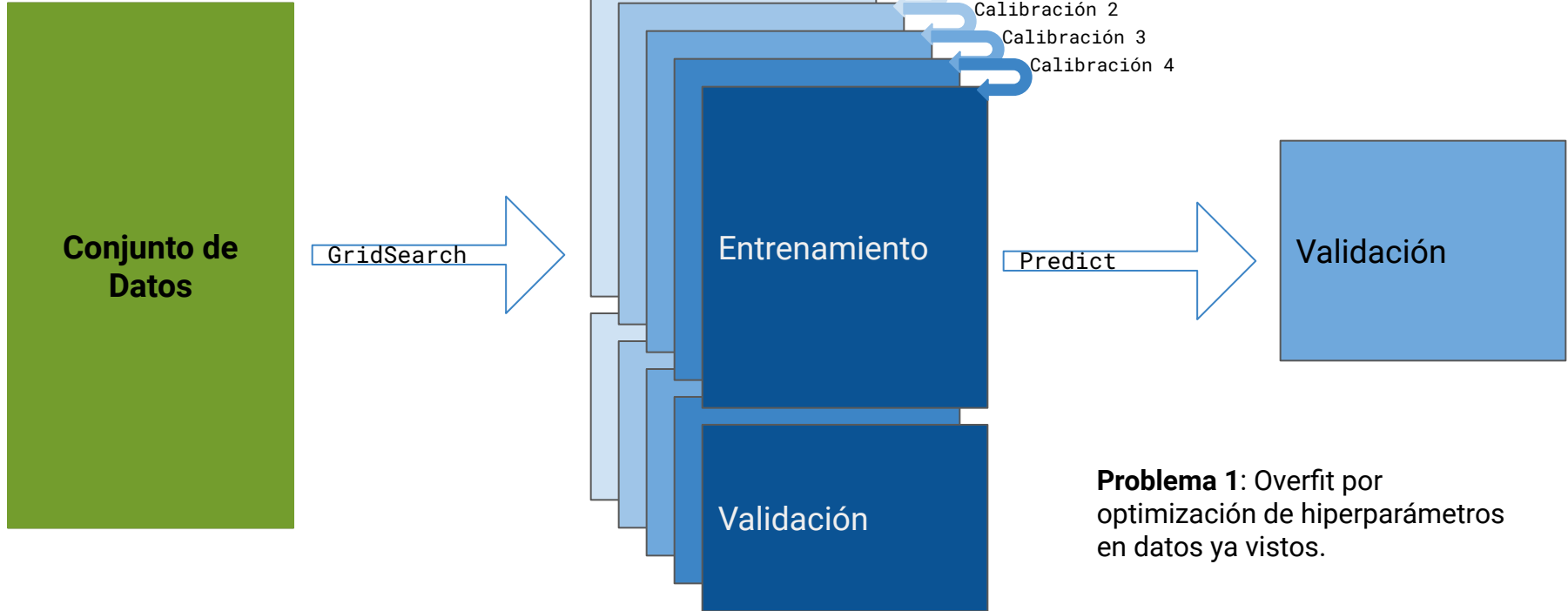
# El escenario



# El escenario



# El escenario



**Problema 1:** Overfit por optimización de hiperparámetros en datos ya vistos.

**Problema 2:** Overfit por aprendizaje de la máquina en datos ya vistos.

# Solución

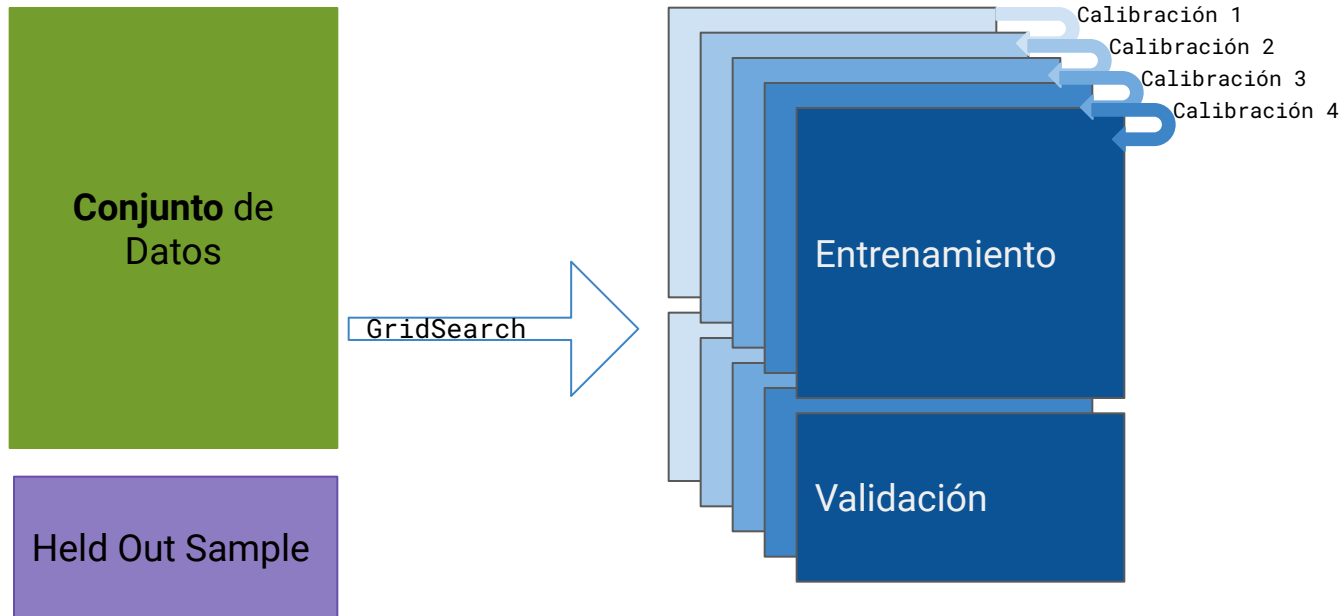


The diagram consists of two stacked rectangles. The top rectangle is olive green and contains the text 'Conjunto de Datos'. The bottom rectangle is purple and contains the text 'Held Out Sample'. This visualizes the partitioning of a dataset into a training set and a held-out sample.

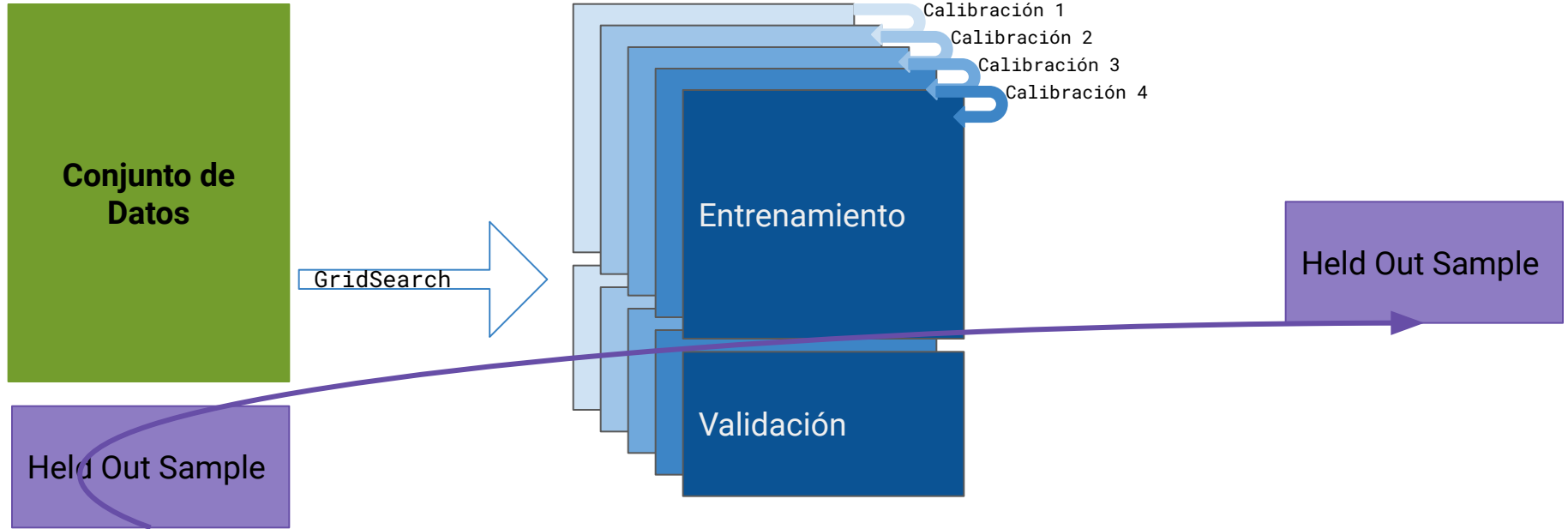
Conjunto de  
Datos

Held Out Sample

# Solución

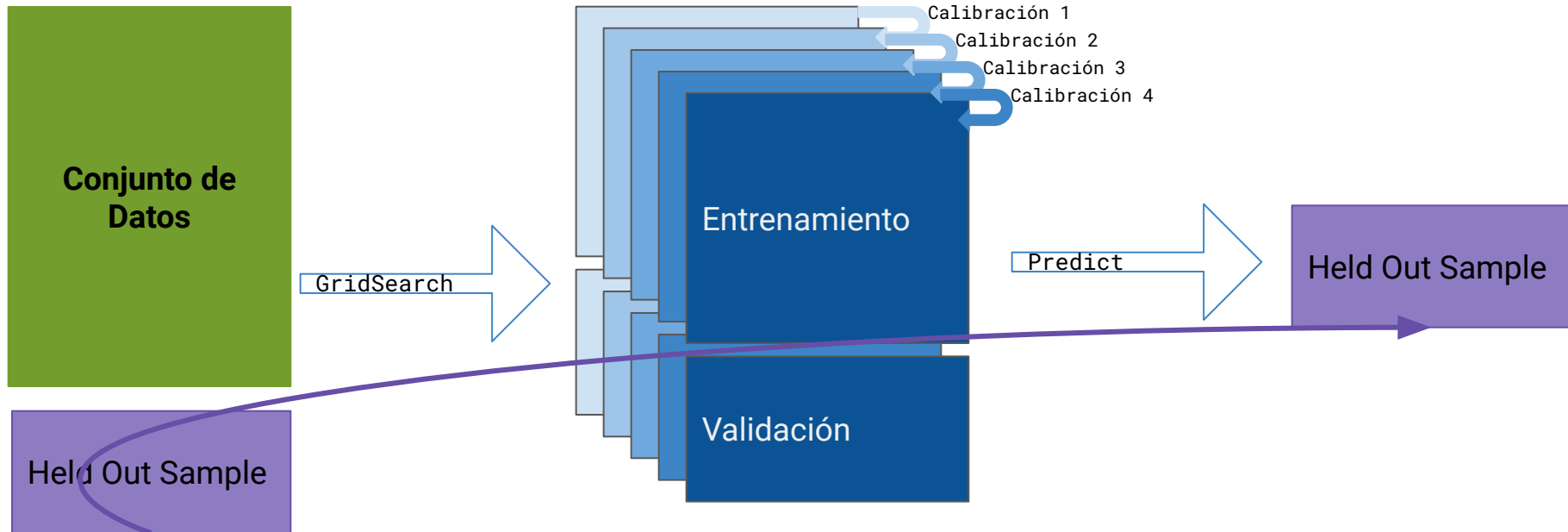


# Solución



Evaluación con datos  
excluidos de la  
Calibración y el  
entrenamiento

# Solución



Evaluación con datos  
excluidos de la  
Calibración y el  
entrenamiento



**{desafío}**  
**latam\_**

*Academia de  
talentos digitales*

[www.desafiolatam.com](http://www.desafiolatam.com)