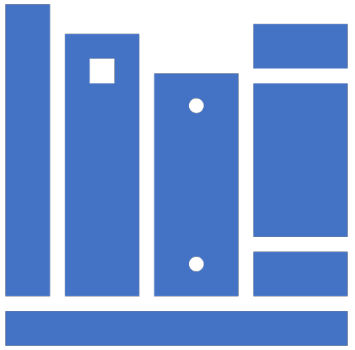


Patrones de Diseño en Machine Learning

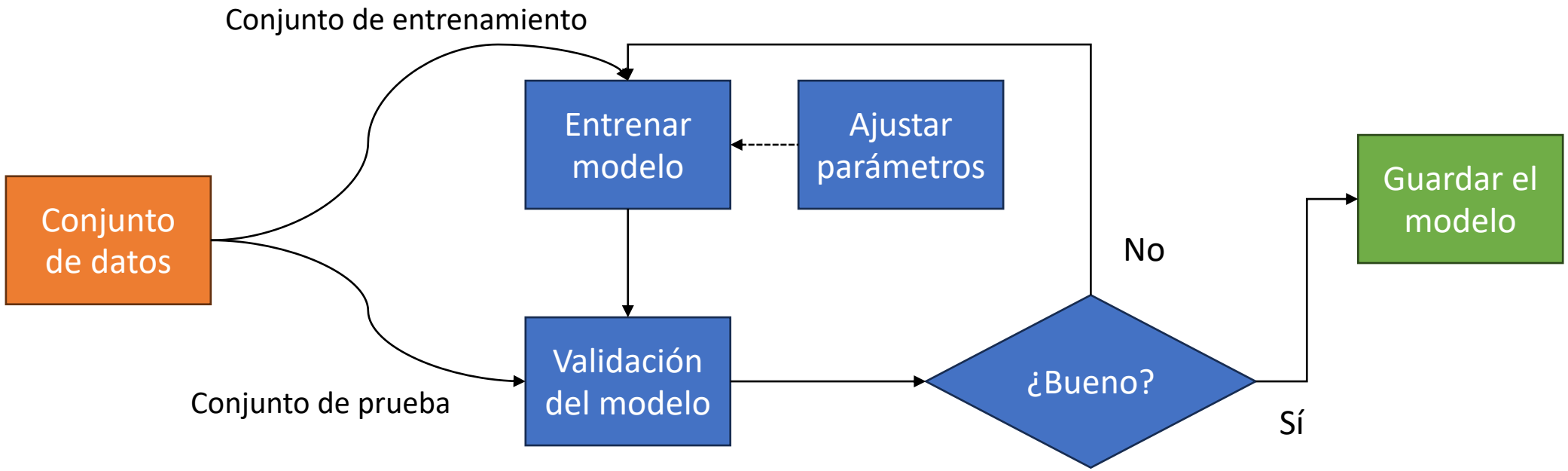
Patrones de Entrenamiento de Modelos

Entrenamiento de modelos de ML

- Formular el problema
- Elegir modelo(s) adecuado(s)
- Analizar y preparar los datos

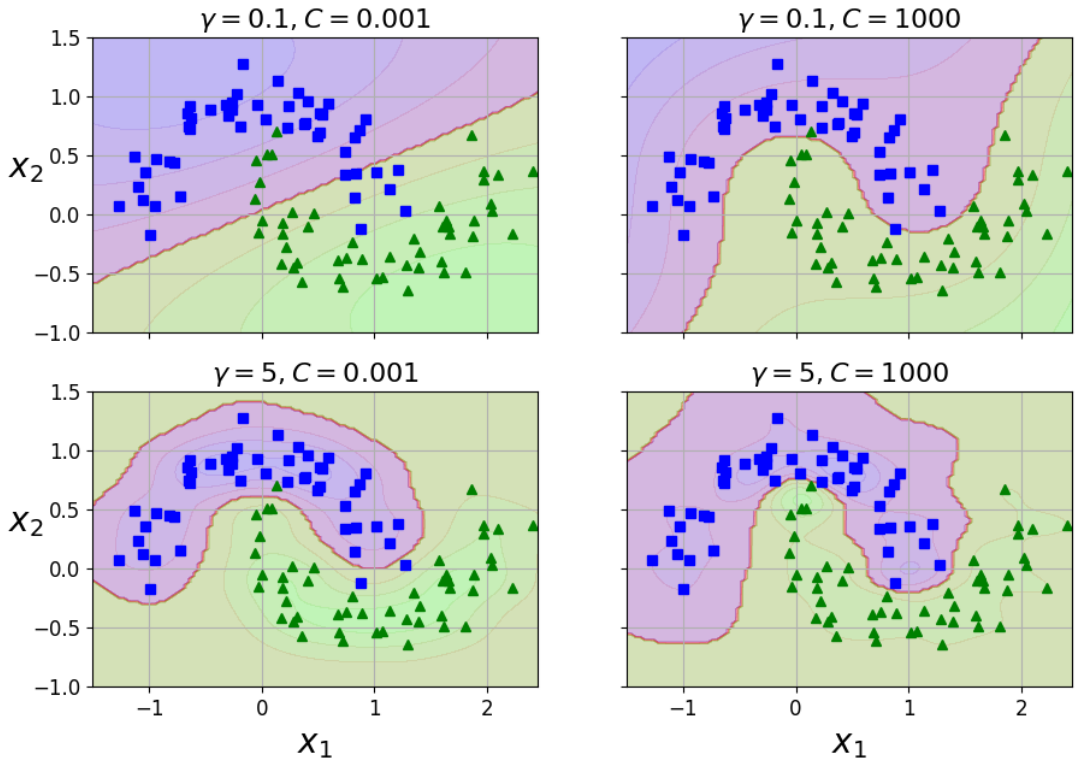
- Entrenar el modelo con validación cruzada
- Ajustar parámetros e hiperparámetros
- Definir métricas y evaluar

- Análisis de error
- Comparar evaluaciones
- Experimentar nuevas ideas
- Guardar y desplegar



Problema

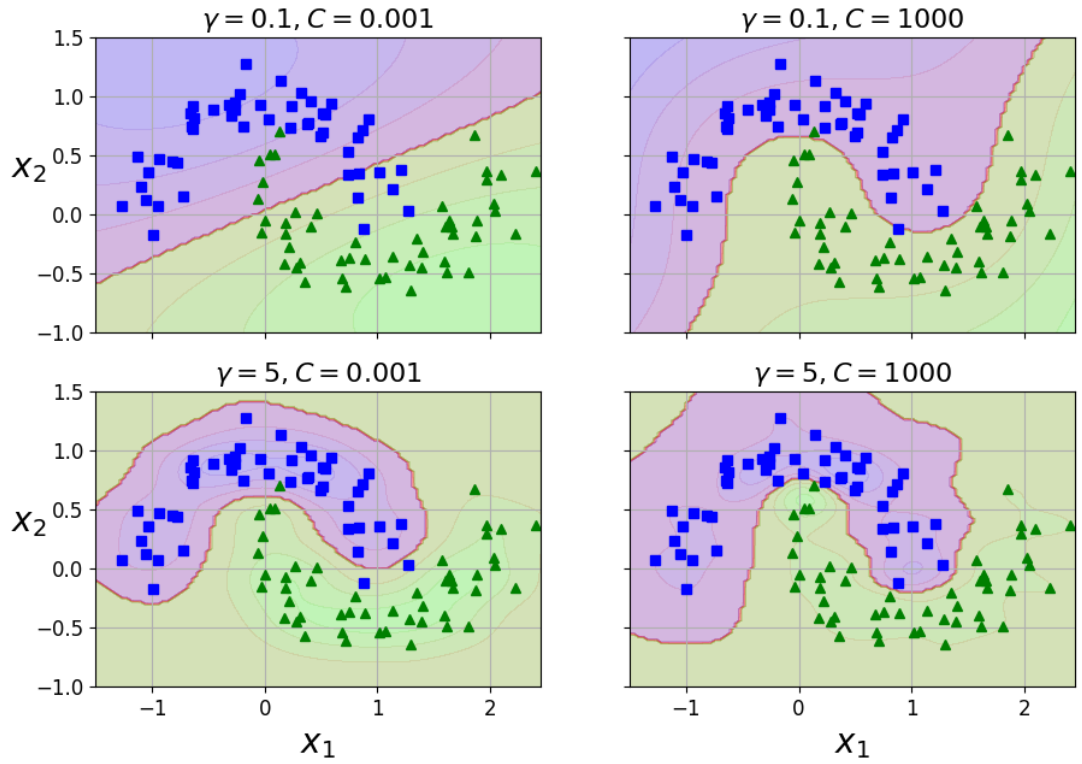
- Uno de los principales pendientes que tenemos al momento de entrenar un modelo es encontrar los mejores parámetros que nos den el mejor resultado.
- Por supuesto, evitando el sobreajuste e infraajuste.



Patrón 8: Ajuste de (hiper)parámetros

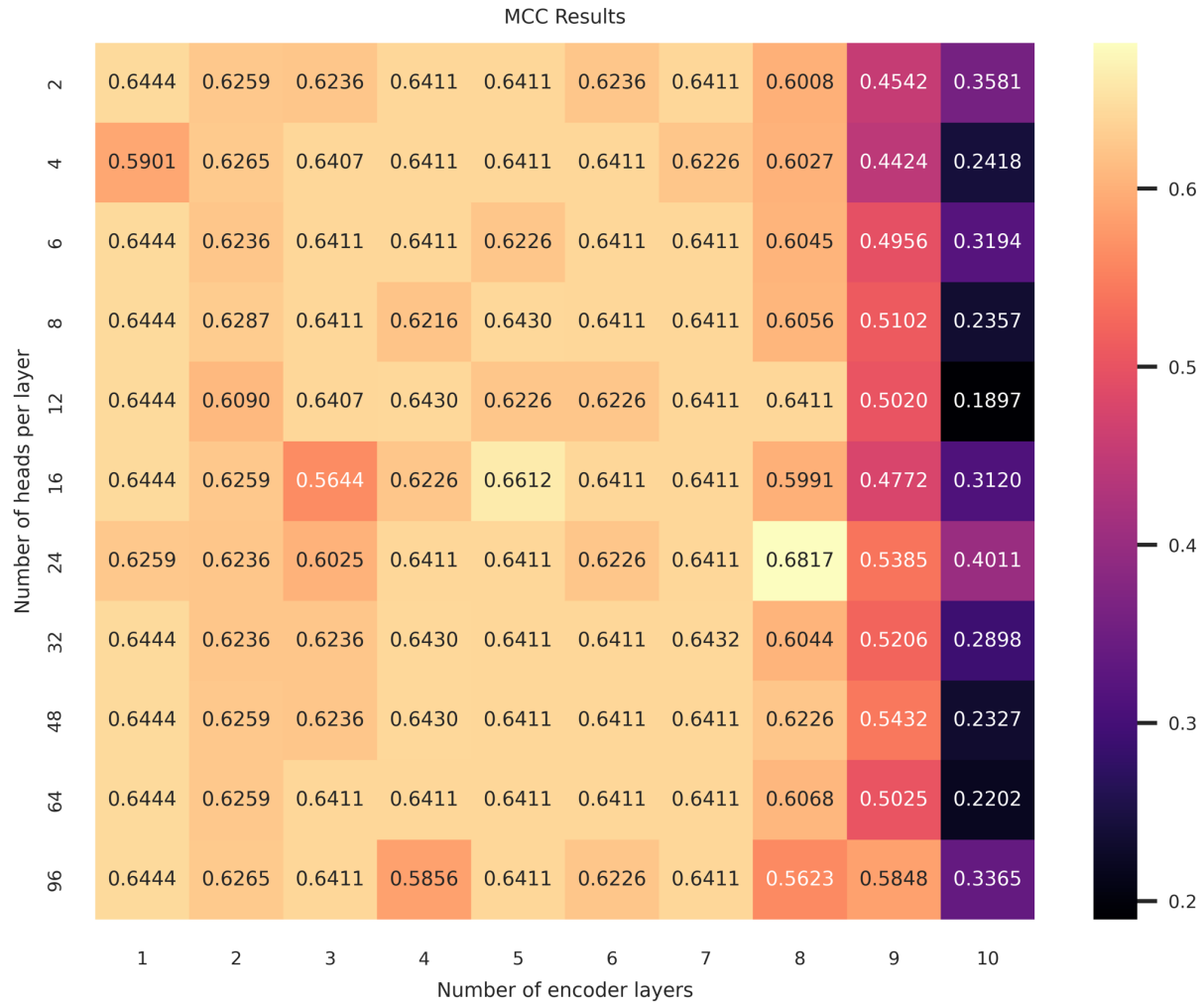
Para ajustar los (hiper)parámetros de un modelo, tenemos diversas opciones:

- Búsqueda «manual»
- Optimización bayesiana



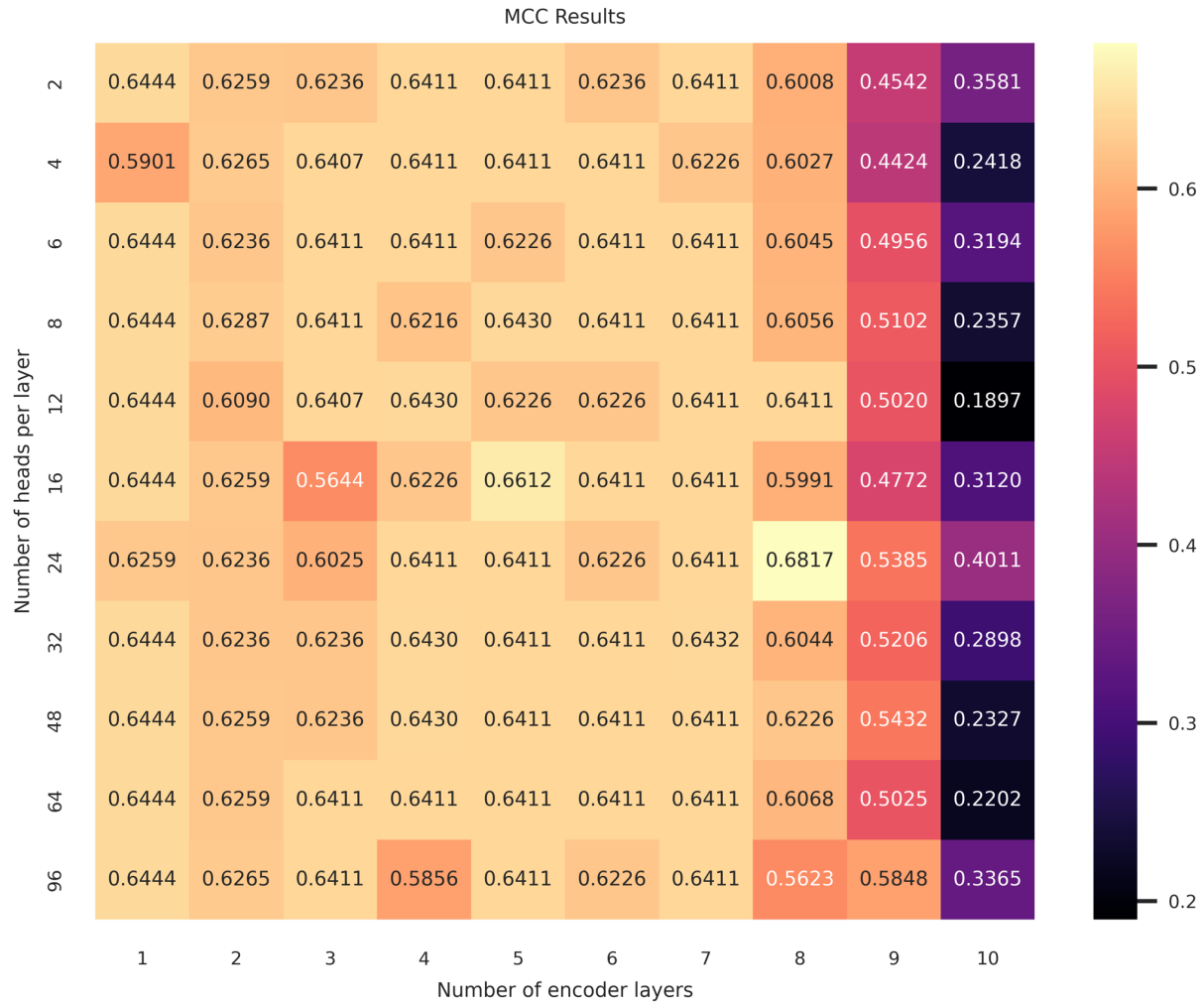
Patrón 8: Ajuste de (hiper)parámetros

- La búsqueda «manual» implica buscar brutamente en un espacio de posibles valores para los parámetros.
- Se suelen llamar mallas de búsqueda.
- Una opción es, en lugar de buscar en todo el espacio, seleccionar solo un subconjunto de valores al azar.



Patrón 8: Ajuste de (hiper)parámetros

- Aunque útil, realizar mallas de búsqueda es *muy* tardado, ya que se entrena un modelo para cada combinación de parámetros posible que se declara.
- Muchos modelos en un espacio amplio, puede tardar hasta días.

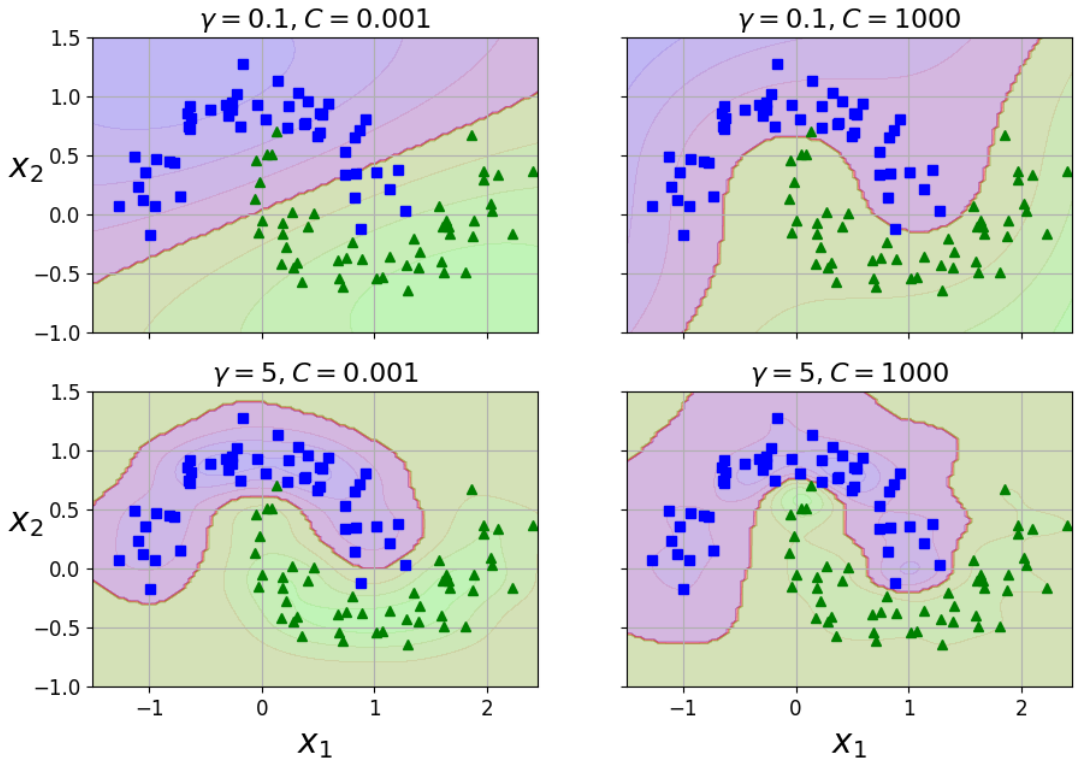


Patrón 8: Ajuste de (hiper)parámetros

Existen varias librerías de código abierto en Python que se encargan de implementar diversas ideas para optimizar parámetros e hiper parámetros.

Para redes neuronales:

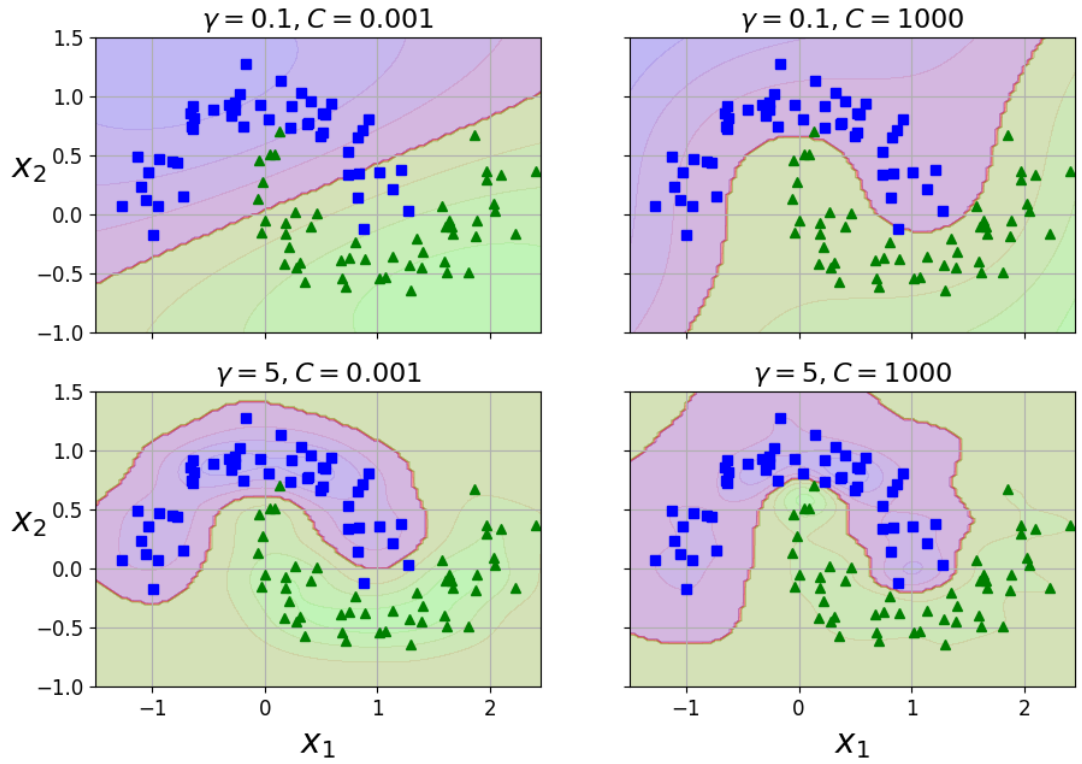
- Hyperopt
- Hyperas
- Keras Tuner



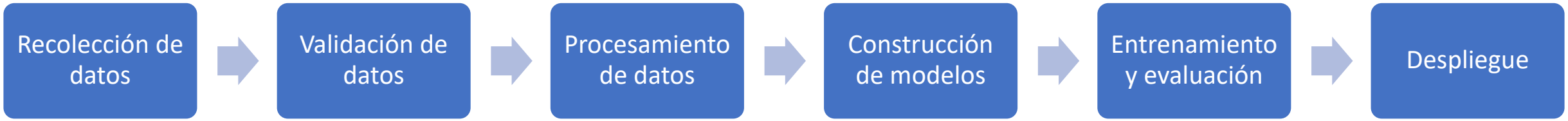
Patrón 8: Ajuste de (hiper)parámetros

Para uso general:

- Scikit-Optimize
- Spearmint
- Hyperband
- Sklearn-Deap



Problema



- Hasta el momento, hemos trabajado con la programación en un único programa o libro de Jupyter ya que es más fácil mantener nuestra programación en orden.
- Debemos tener cuidado en los procesos, ya que estos deben aplicarse de la misma manera para nuestros datos o para el entrenamiento/validación de los modelo.
- Por ejemplo, una transformación de los datos se entrena con el conjunto de entrenamiento, debe aplicarse a los datos para la validación.

Patrón 9: Pipelines

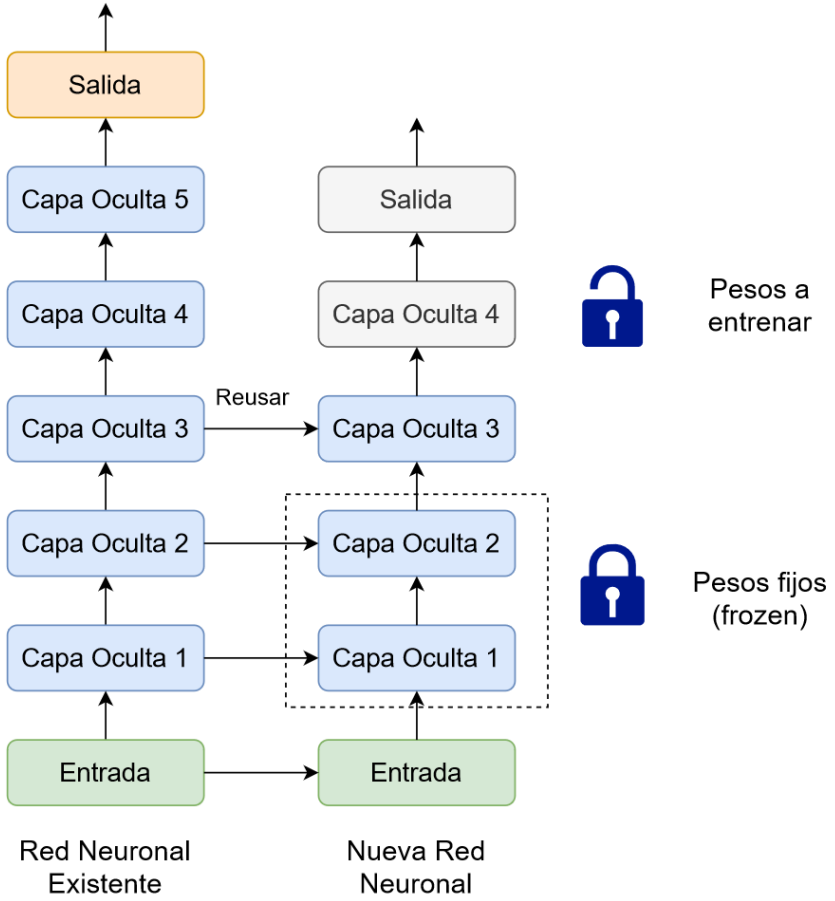
- La idea un pipeline es dividir cada parte del código en pequeños pedazos.
- Esto nos permite manejar partes del proceso más pequeñas e independientes.
- Además de que nos enfocamos en la reproducibilidad de los métodos y resultados.

Patrón 9: Pipelines

- No todo debe ser una parte del pipeline, pero ciertamente es buena idea.
- Scikit-learn, Keras y Pytorch tienen sus propios métodos para crear esta abstracción de los procesos.

Patrón 10: Transfer Learning

- Este patrón es propio de redes neuronales.
- Es MUY conveniente, ya que permite entrenar modelos pre entrenados con nuestros datos para diversas tareas rápidamente.
- Usualmente empleado con modelos que usan imágenes y texto.



Patrón 11: Reproducibilidad

- Algo muy importante es controlar el factor aleatorio de diversos procesos del sistema.
- Siempre se debe usar semillas para tales procesos.
- Recuerden cambiarla de vez en cuando.




Final de la presentación

¡Gracias por su atención!