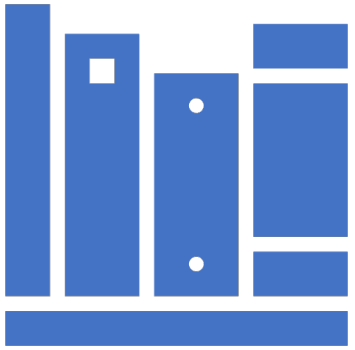


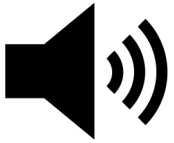
Patrones de Diseño en Machine Learning

Patrones de Representación de Problemas

Introducción

- En la presentación anterior cubrimos las diversas formas en las que las entradas de un modelo de ML se pueden representar.
- Ahora analizaremos diversos problemas de ML y como afectan las arquitecturas de los modelos.

Introducción



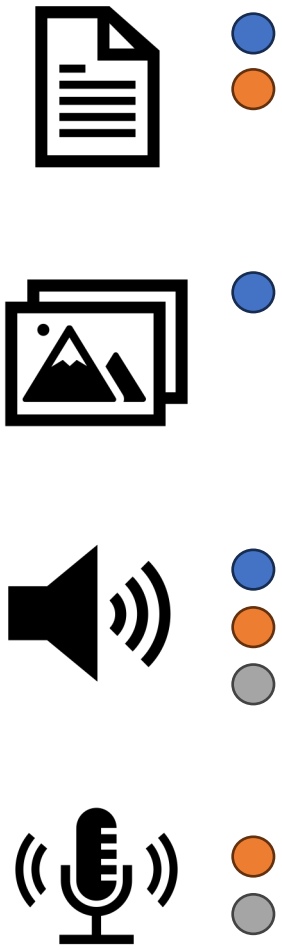
?

Redes neuronales convolucionales

Redes neuronales recurrentes

Series de tiempo

Introducción



Son arquitecturas muy especializadas (Deep Learning), vamos a enfocarnos en cosas más generales.

Redes neuronales convolucionales

Redes neuronales recurrentes

Series de tiempo

Problema

El primer paso para construir un modelo de ML es formular el problema.

- ¿Supervisado o no supervisado?
- ¿Regresión o clasificación?
- ¿Cuáles son las características?
- Si es supervisado, ¿cuáles son las etiquetas?

Problema

Consideremos el caso de predecir cuánta lluvia caerá en los próximos 15 minutos en una ubicación en particular.

- ¿Regresión o clasificación?

Problema

Consideremos el caso de predecir cuánta lluvia caerá en los próximos 15 minutos en una ubicación en particular.

- ¿Regresión o clasificación?
- Dado que es una cantidad (e.g., 0.3 cm), y esta puede depender de datos históricos, puede ser buena idea usar una serie de tiempo.

Problema

Consideremos el caso de predecir cuánta lluvia caerá en los próximos 15 minutos en una ubicación en particular.

- Otra opción es simplemente considerar la predicción numérica, es decir, un problema de regresión.

Problema

Supongamos que entrenamos el modelo y, para sorpresa de todos, fallamos miserablemente.

- Predecir el clima es muy difícil.
- Para cierta combinación de características, tenemos que llueve 0.3 o 0.5 cm.
- ¿Qué podemos hacer para mejorar?

Problema

La caída de lluvia es probabilística.

- Para el mismo conjunto de características, a veces llueve 0.3 cm o 0.5 cm.
- A pesar de entrenar el modelo con **ambos** escenarios, este solo arroja **uno solo**.
- ¡Vamos a cambiar la formulación del problema!

Patrón 3: Reframing

Vamos a cambiar el problema de regresión a uno de clasificación.

- Una forma es modelar una distribución de probabilidad discreta.
- Ahora tenemos un problema de clasificación multiclase donde las etiquetas son los rangos de precipitación.

Precipitación (mm)	Salida
0 – 0.5	0.009
0.5 – 1.0	0.01
1.0 – 1.5	0.07
1.5 – 2.0	0.85
2.0 – 2.5	0.05
2.5 – 3.0	0.01
3.0 +	0.001

Patrón 3: Reframing

Funciona porque…

Cambiar de una representación a otra es útil cuando el cambio permite obtener una ventaja.

- Supongamos que queremos un sistema de recomendación de videos.
- Puede ser un problema de clasificación ya que buscamos si el usuario ve el video o no.
- ¿Notan algún problema?

Patrón 3: Reframing

Funciona porque…

Este enfoque puede generar abuso por parte de los usuarios:

- Si entienden lo que hace el modelo, pueden priorizar ciertas características.
- Es decir, **click bait**.

Patrón 3: Reframing

Funciona porque…

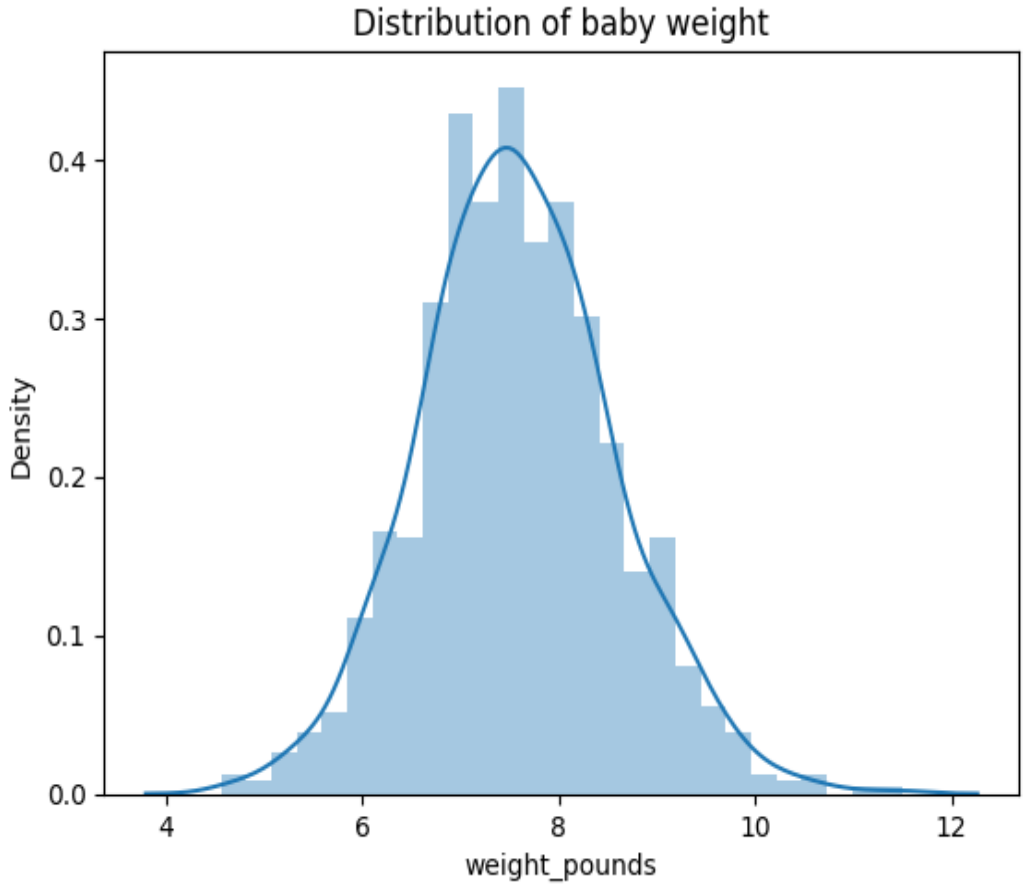
Si cambiamos a un problema de regresión…

- Buscamos predecir la fracción del video que el usuario verá.

Patrón 3: Reframing

Funciona porque…

Captura la incertidumbre presente en los datos, que se refleja en la apertura o desviación estándar presenten en los datos.

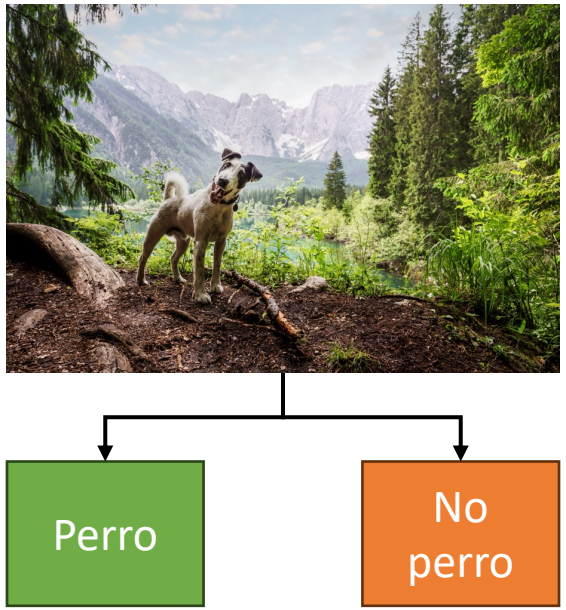


Patrón 3: Reframing

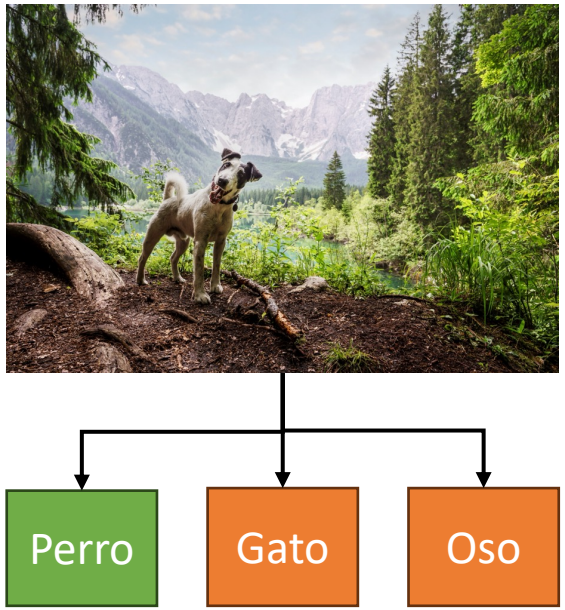
Vamos a Google Colab para explorar más sobre cómo funciona y sus desventajas…

Problema

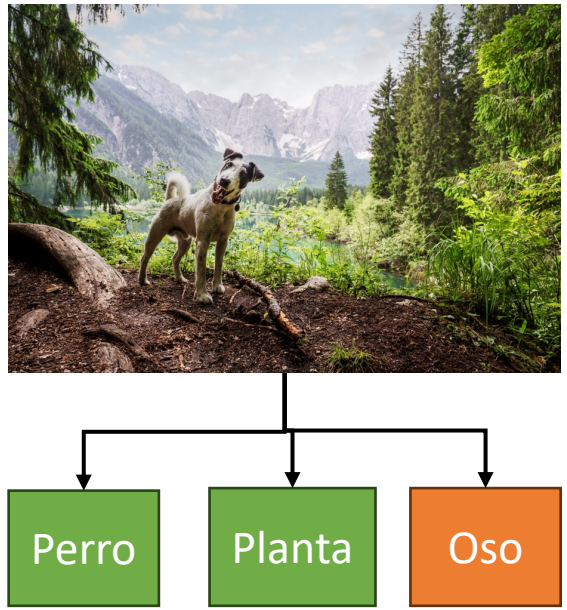
Clasificación Binaria



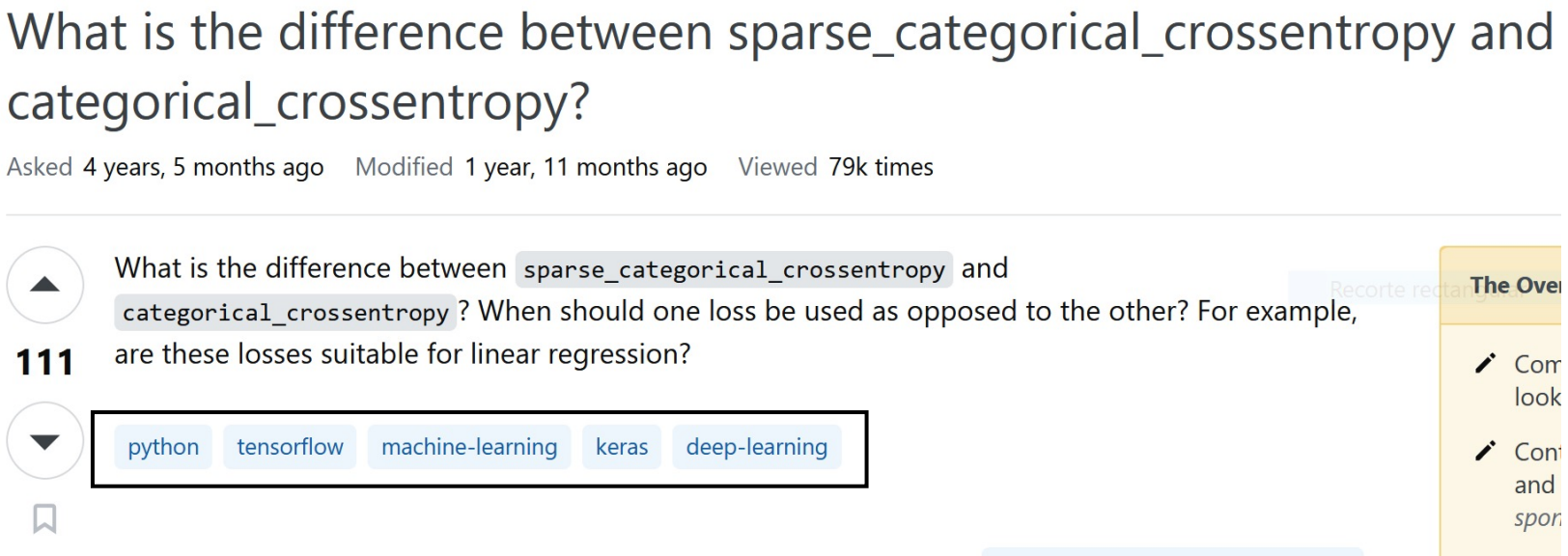
Clasificación Multiclase



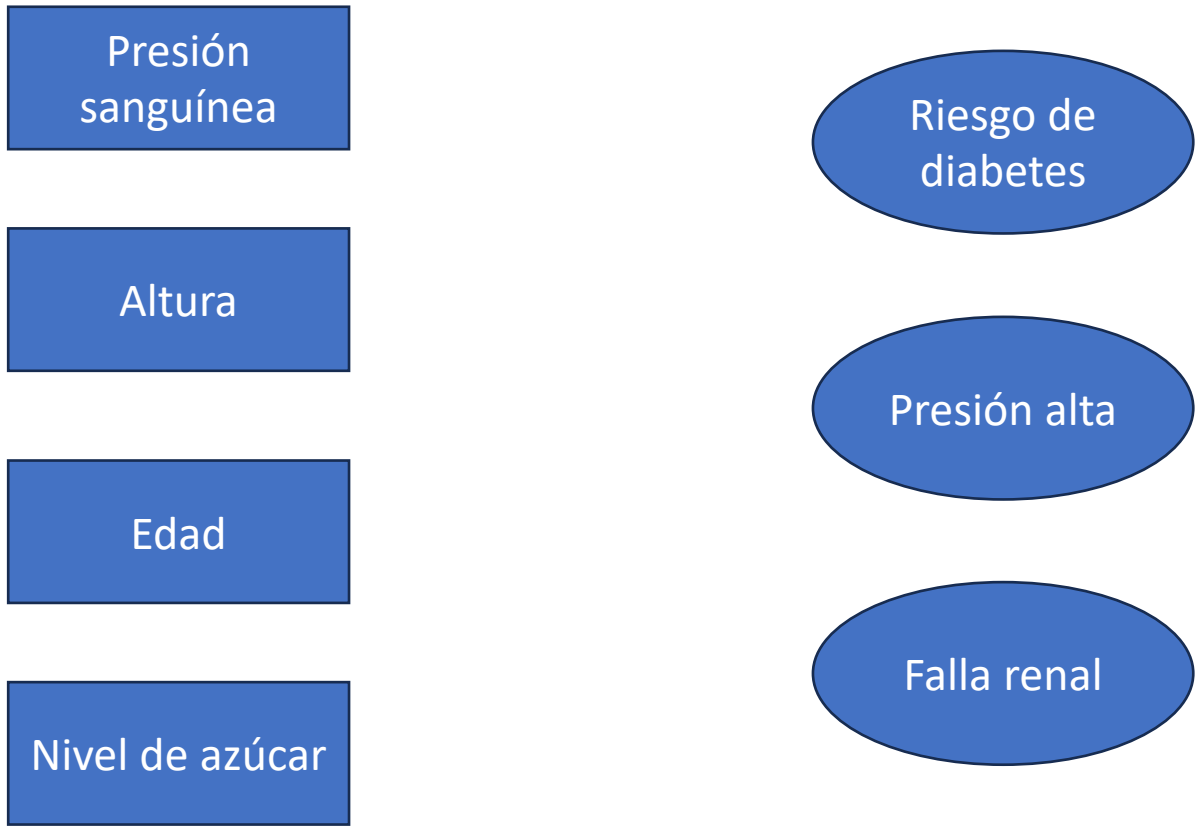
Clasificación Multietiqueta



Problema

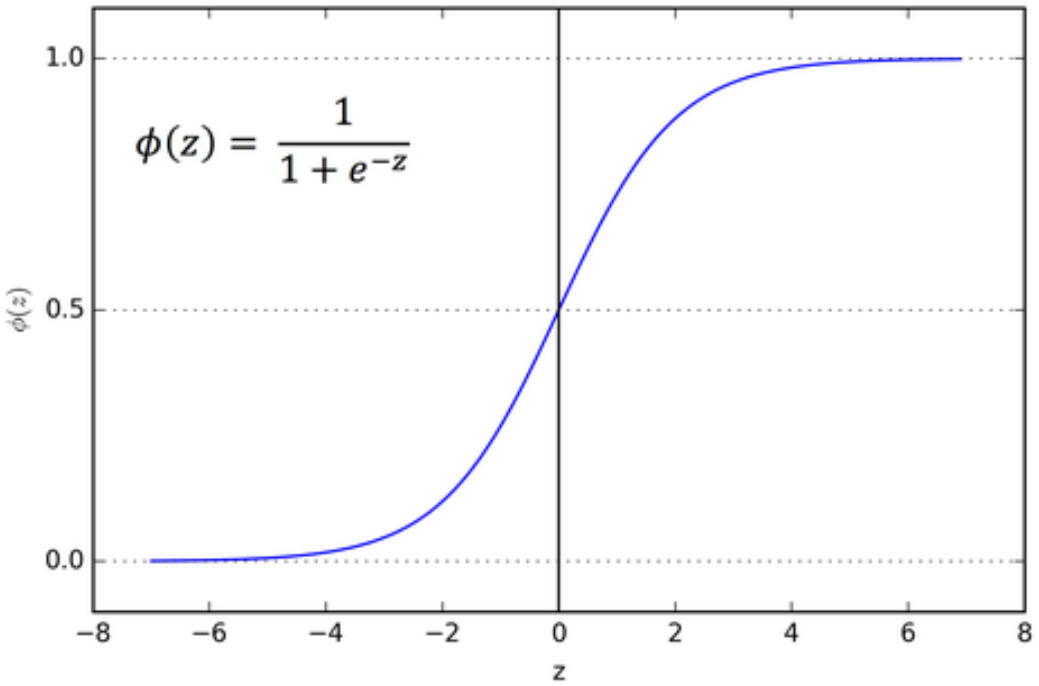

What is the difference between sparse_categorical_crossentropy and categorical_crossentropy?
Asked 4 years, 5 months ago
Modified 1 year, 11 months ago
Viewed 79k times
What is the difference between sparse_categorical_crossentropy and categorical_crossentropy? When should one loss be used as opposed to the other? For example, are these losses suitable for linear regression?
111
python
tensorflow
machine-learning
keras
deep-learning

Problema


Presión sanguínea
Altura
Edad
Nivel de azúcar
Riesgo de diabetes
Presión alta
Falla renal

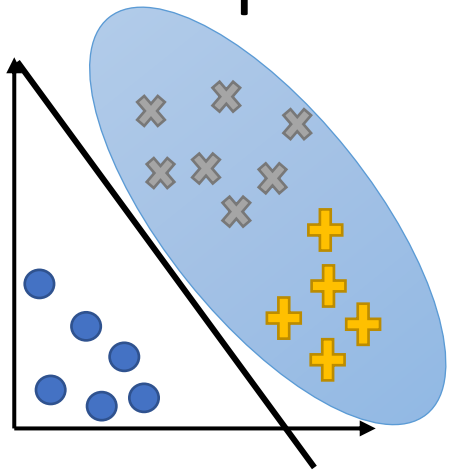
Patrón 4: Clasificación Multietiqueta

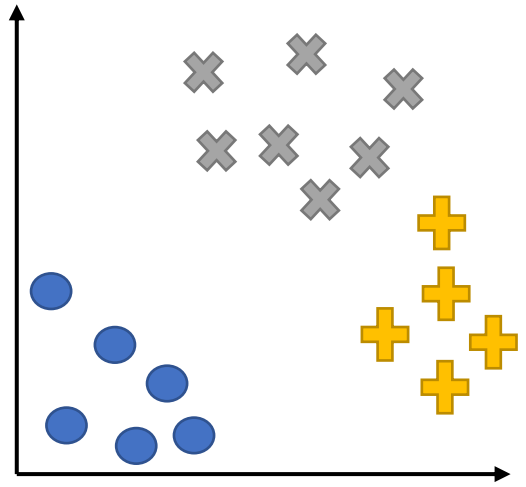
- Diseñar modelos de ML que puedan predecir más de una clase al mismo tiempo.
- Comprende **redes neuronales** donde la función de activación de la capa de salida es la **función sigmoide**.
- Cada neurona de salida saca un valor entre 0 y 1, en lugar de que todas sumen 1 (cuando se usa *softmax*).

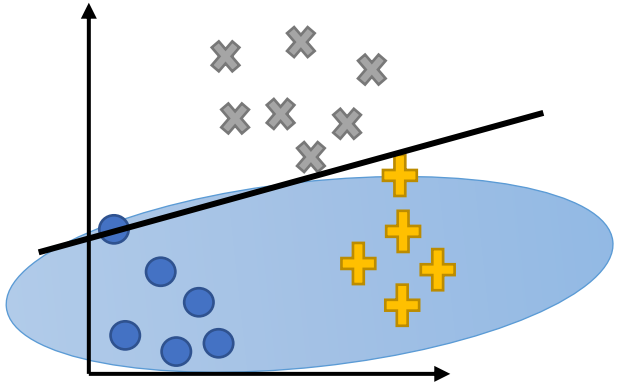


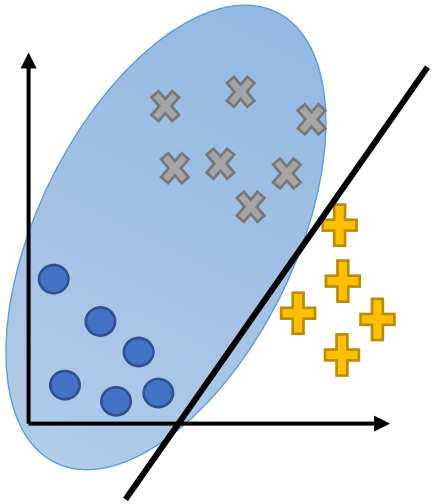
Patrón 4: Clasificación Multietiqueta

- Otra opción es usar clasificación one vs rest.
- Se entrena un modelo para detectar si una clase de las n posibles se activa o no.
- Todo esto para cada posible clase.







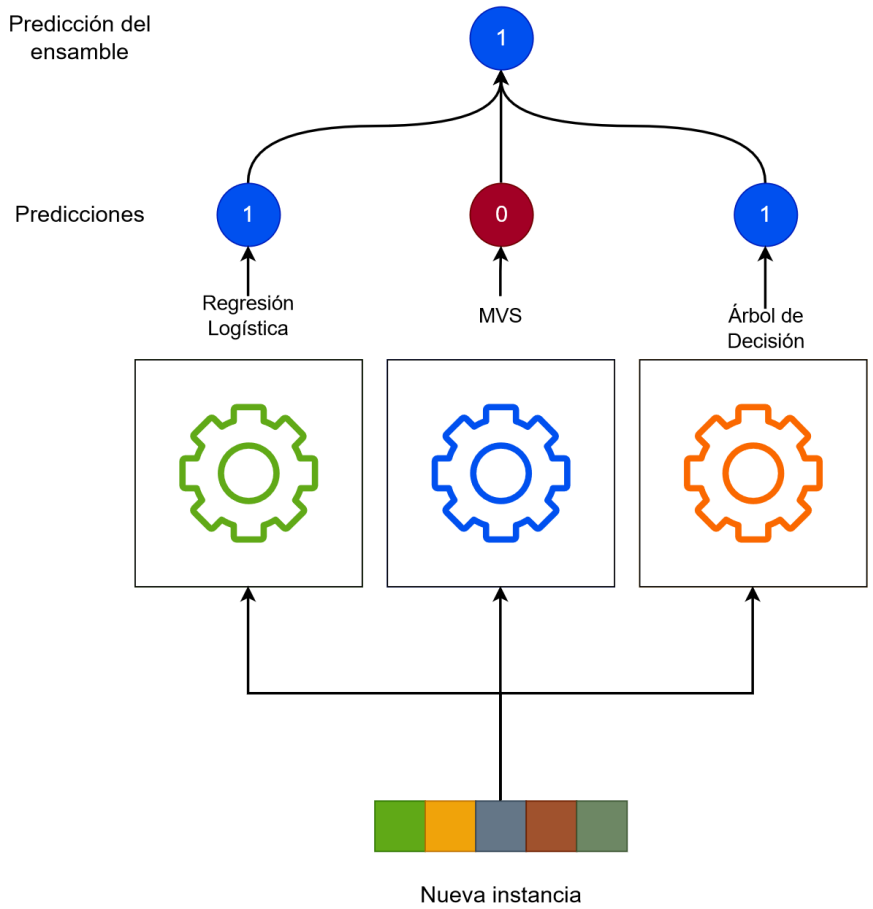


Patrón 4: Clasificación Multietiqueta

Vamos a Google Colab para poner en práctica lo aprendido…

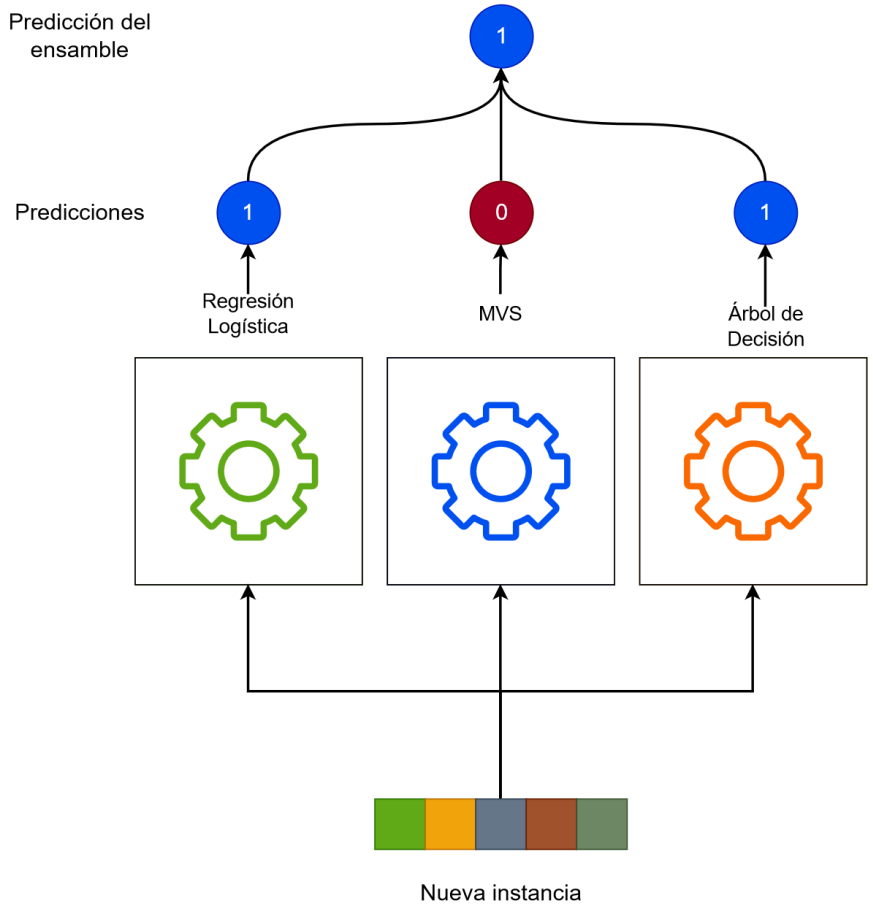
Patrón 5: Ensambles

- Nosotros ya sabemos la historia de los métodos de Ensambles, porqué funcionan y sus ventajas.
- No necesitamos repasar todo otra vez.
- Solo vamos a mencionar sus desventajas.



Patrón 5: Ensambles

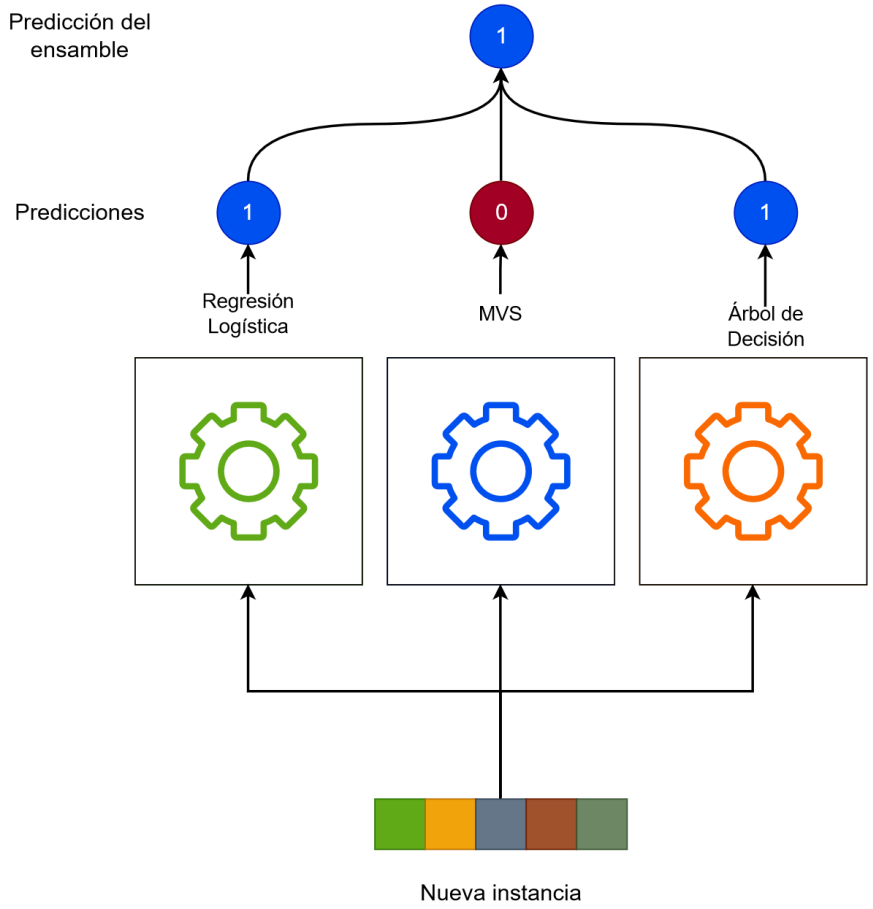
- En práctica existen dos familias de Ensambles que se usan ampliamente:
  - Bagging, en particular Bosques Aleatorios.
  - Boosting, en particular XGBoost, CatBoost.



Patrón 5: Ensambles

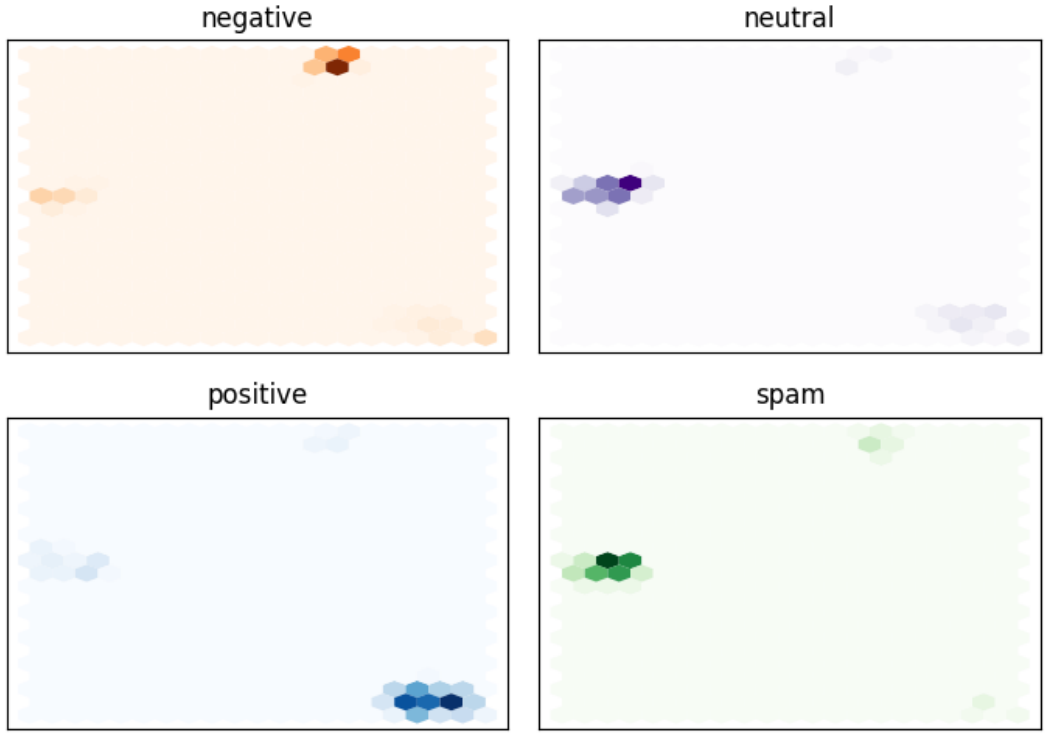
Entre sus desventajas se encuentran:

- Mayor tiempo de entrenamiento y diseño.
- Menor interpretabilidad del modelo final.
- Es necesario elegir la herramienta adecuada.
  - Alto sesgo → Boosting
  - Alta varianza → Bagging



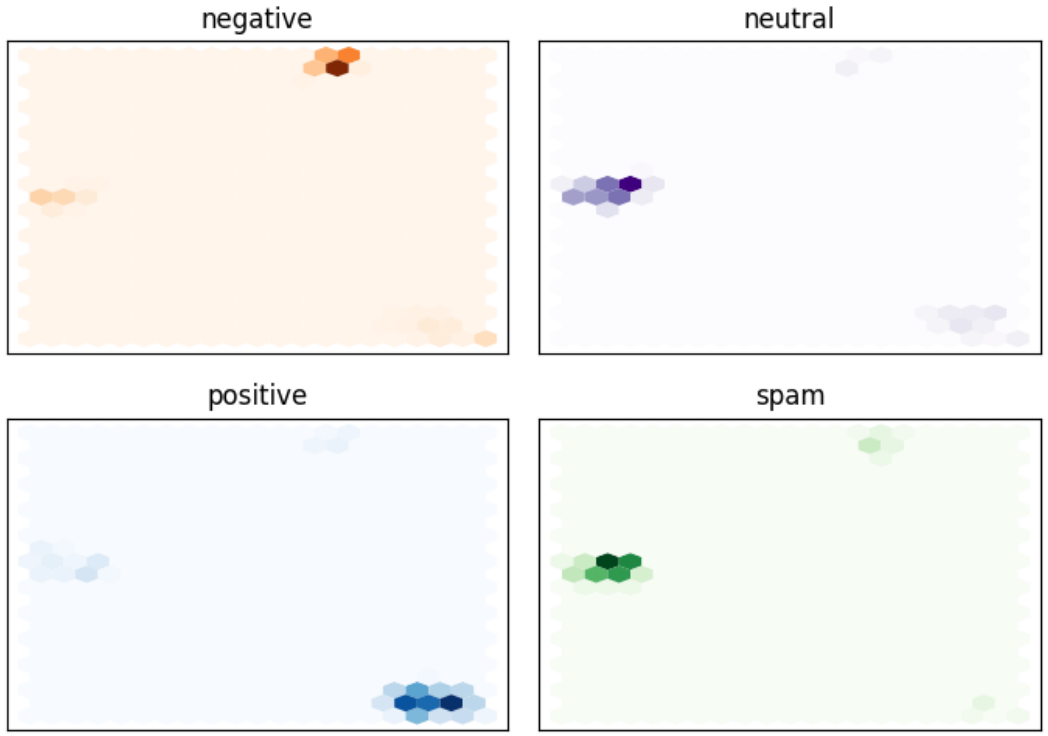
Patrón 6: Clase Neutral

- El problema base para clasificación considera dos clases.
- Regresando al problema de los tuits, ¿Por qué considerar únicamente dos clases?
- La idea de agregar más clases es capturar casos especiales para mejorar nuestros modelos.



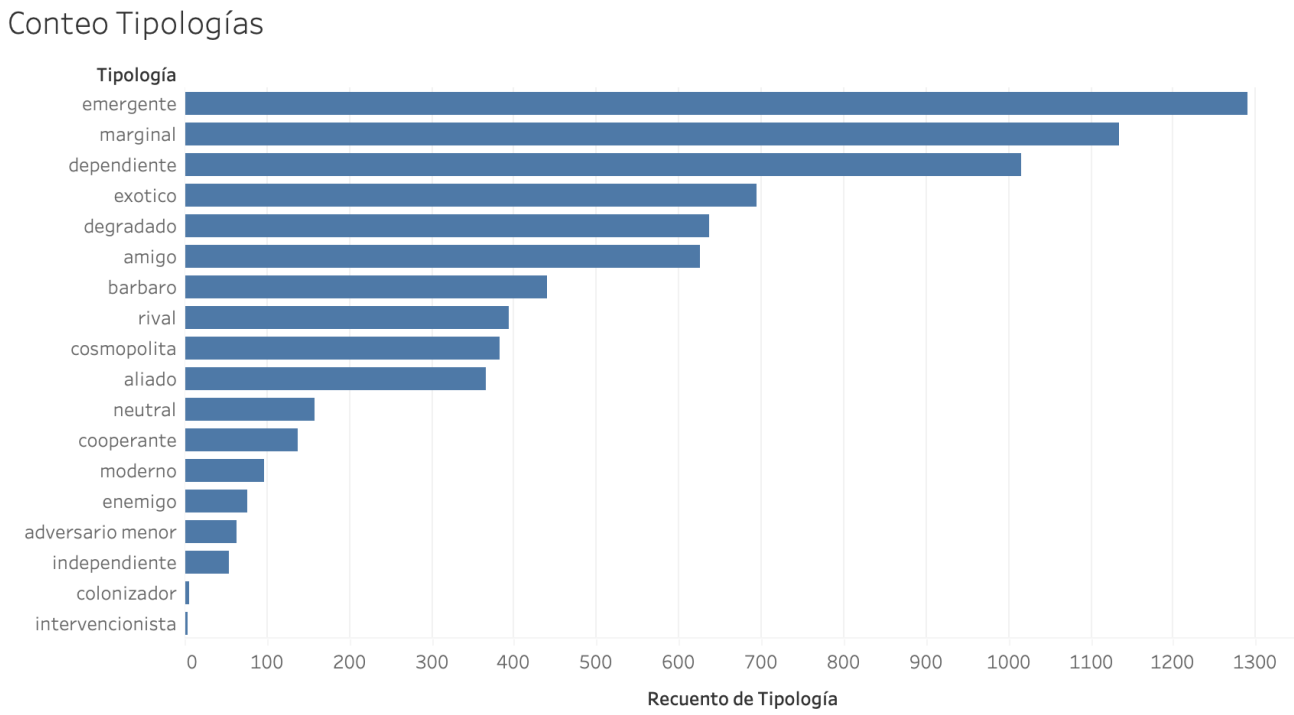
Patrón 6: Clase Neutral

- Esto requiere una planeación desde el principio.
- Ya que necesita que los datos se anoten de manera apropiada.



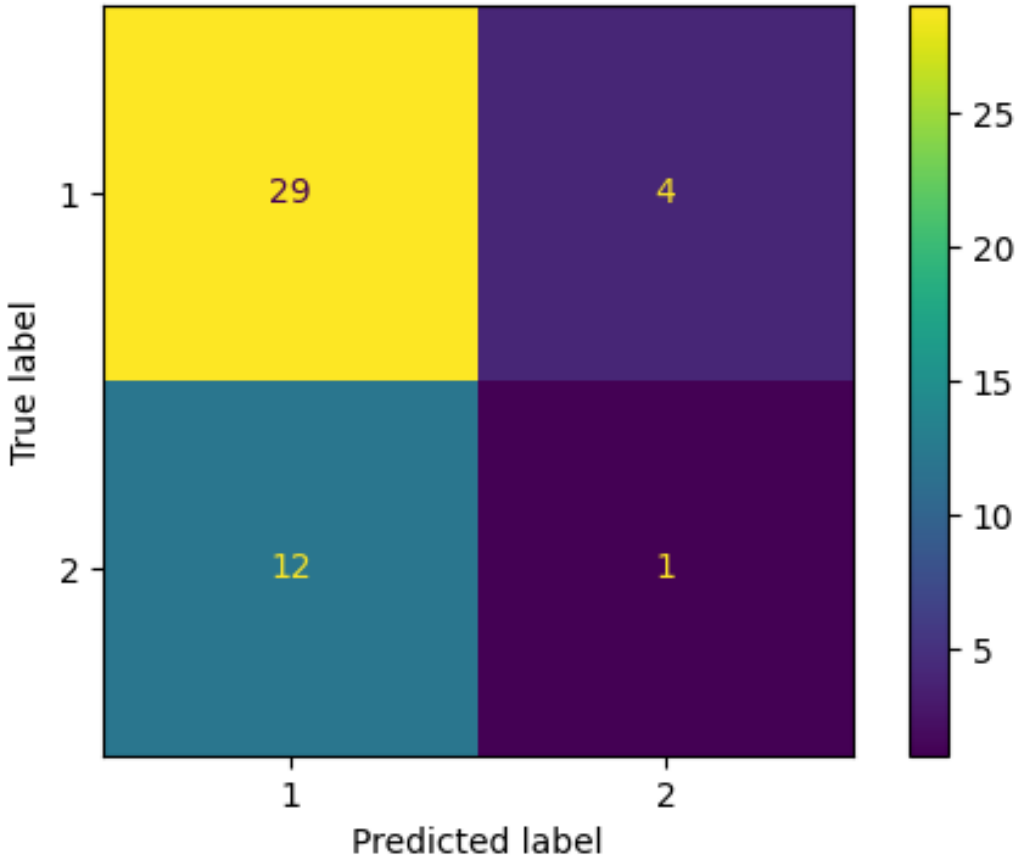
Problema

- Los modelos de aprendizaje suelen aprenden mejor cuando se les proporciona **un número similar de ejemplos** para cada clase durante el entrenamiento.
- Sin embargo, muchos problemas de la vida real **se alejan de este ideal**.
- En el caso de regresión, el desbalance se da cuando se tienen anomalías que se alejan de la median de los datos.



Problema

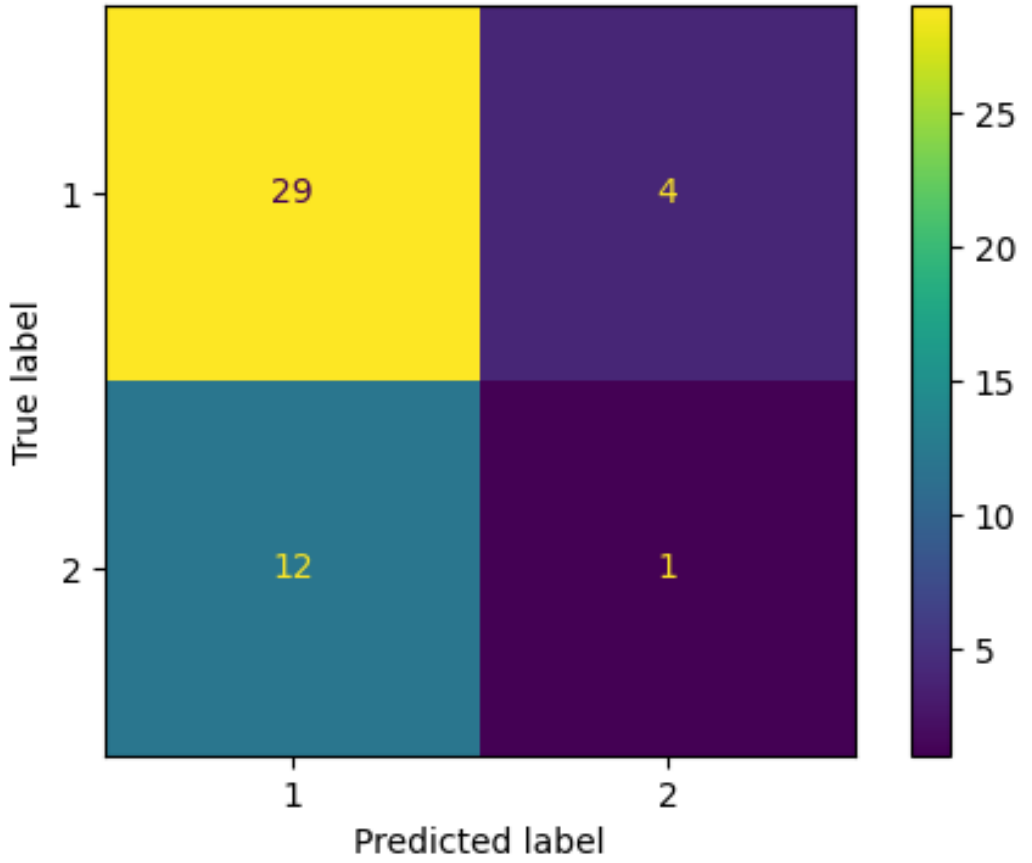
- Como hemos mencionado, no es confiable usar la métrica de *accuracy* como único indicador.
- Proporciona estimaciones infladas (que técnicamente puede ser validas).
- Lo mejor es observar la matriz de confusión.
- El modelo predice la clase mayoritaria mejor que la minoritaria.



Patrón 7: Reequilibrar

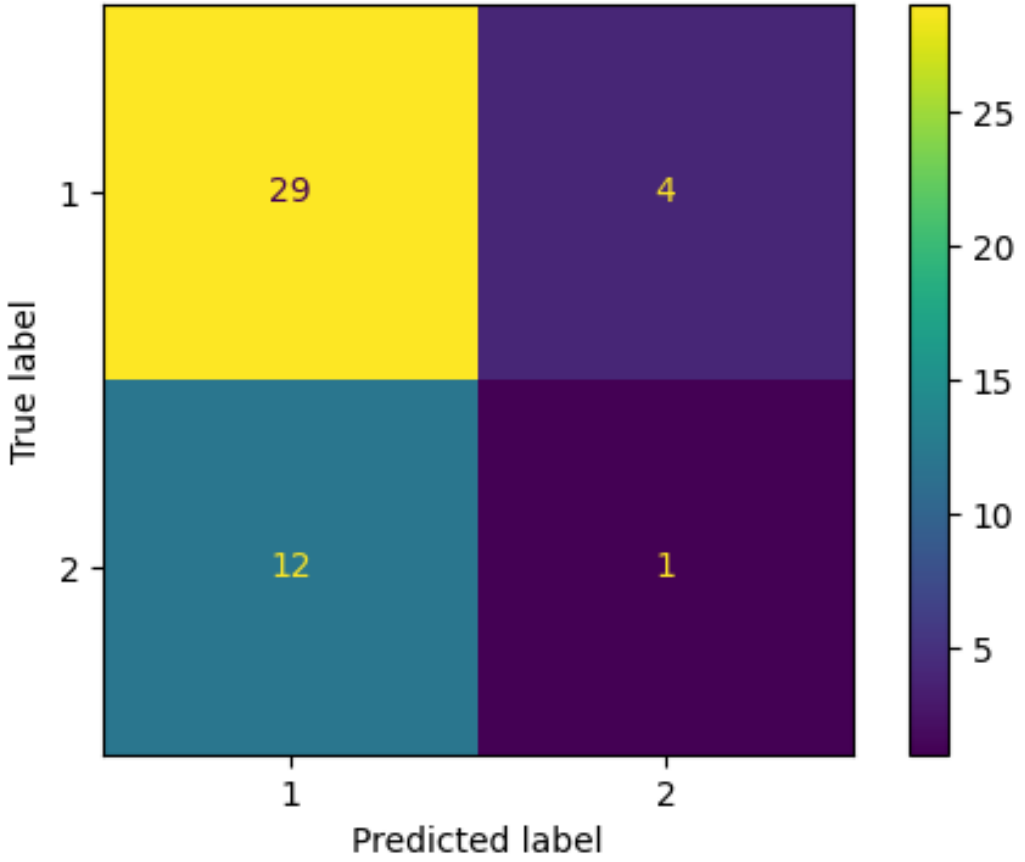
Lo primero que tenemos que hace es elegir buenas métricas de evaluación que reflejen este tipo de situaciones.

- Medida F1
- Precisión balanceada
- Coeficiente de correlación de Matthews.



Patrón 7: Reequilibrar

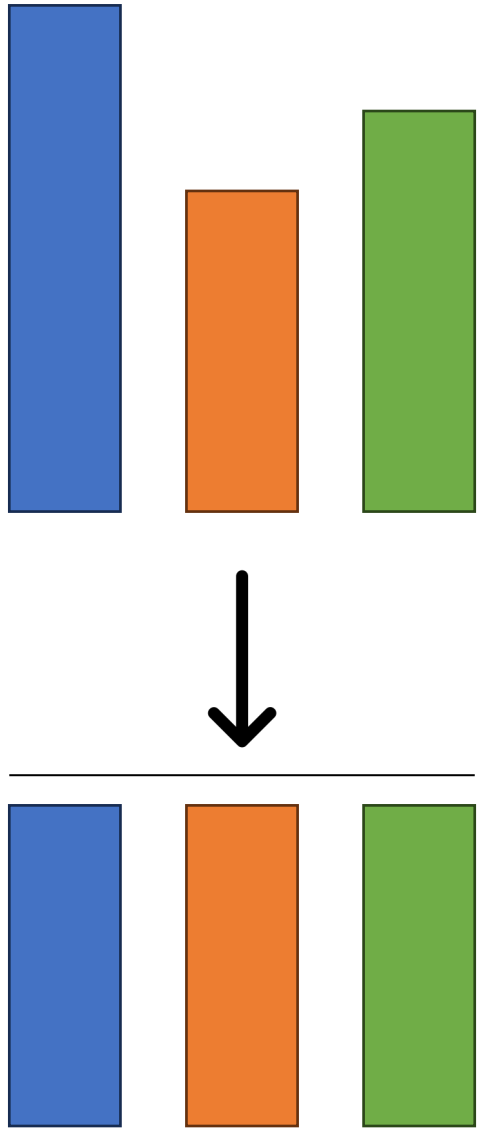
Noten que, antes de aplicar cualquier técnica para solucionar este problema, deben evaluar el rendimiento del modelo con los datos TAL COMO SON.



Patrón 7: Reequilibrar

Downsampling

- La primera familia de ideas recurre a **modificar los datos en lugar del modelo**.
- Con downsampling, vamos **a quitar datos de la clase mayoritaria** hasta encontrar un mejor balance con la minoritaria.



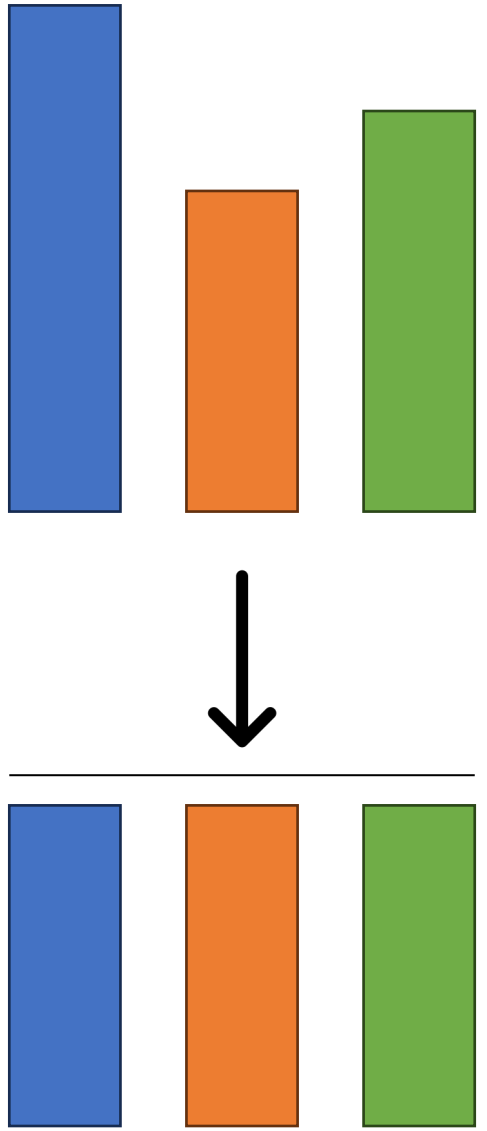
Antes

Después

Patrón 7: Reequilibrar

Downsampling

- Noten que no es necesario llegar al 50/50. Puede ser suficiente algo como 25/75. Es un parámetro que debemos ajustar.
- Esta técnica se suele acompañar con algún Ensamble:
  - Ya que estamos obteniendo nuevos subconjuntos de datos…
  - Se debe hacer manual.



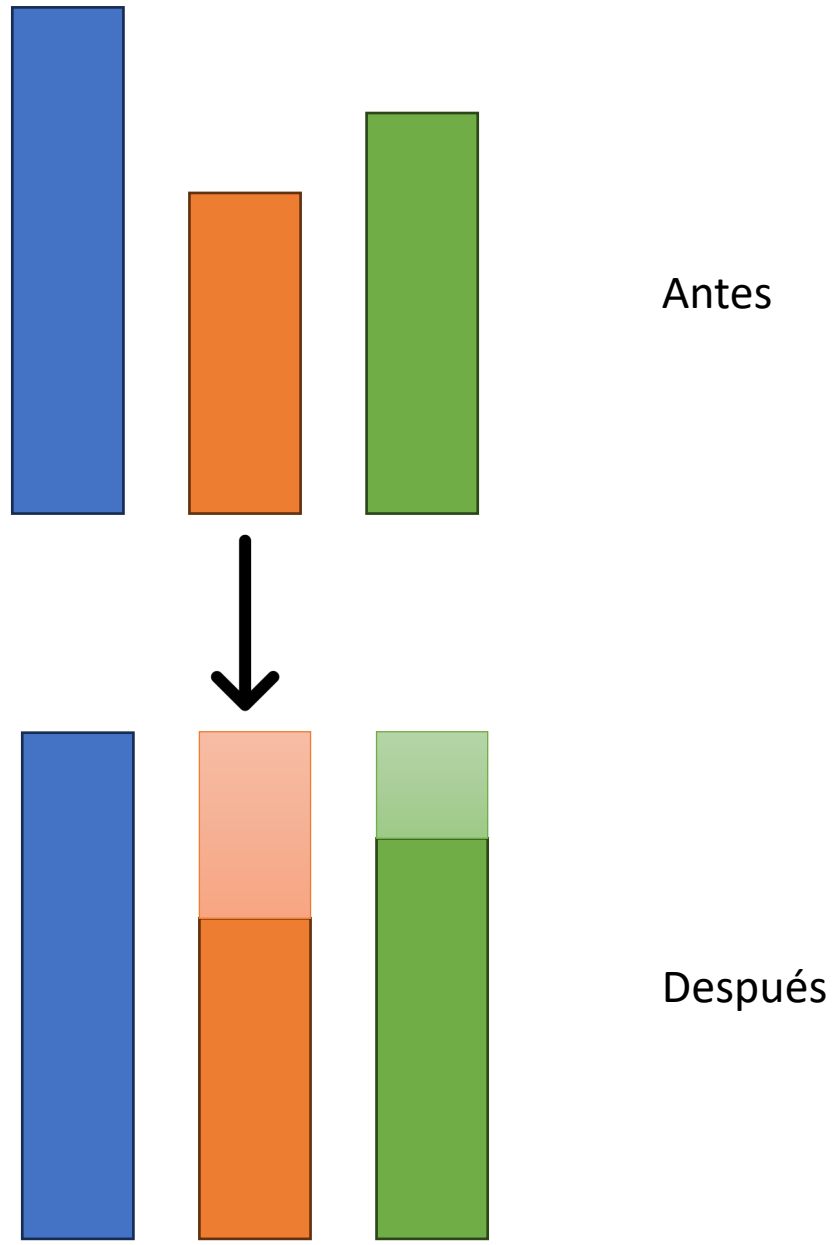
Antes

Después

Patrón 7: Reequilibrar

Upsampling

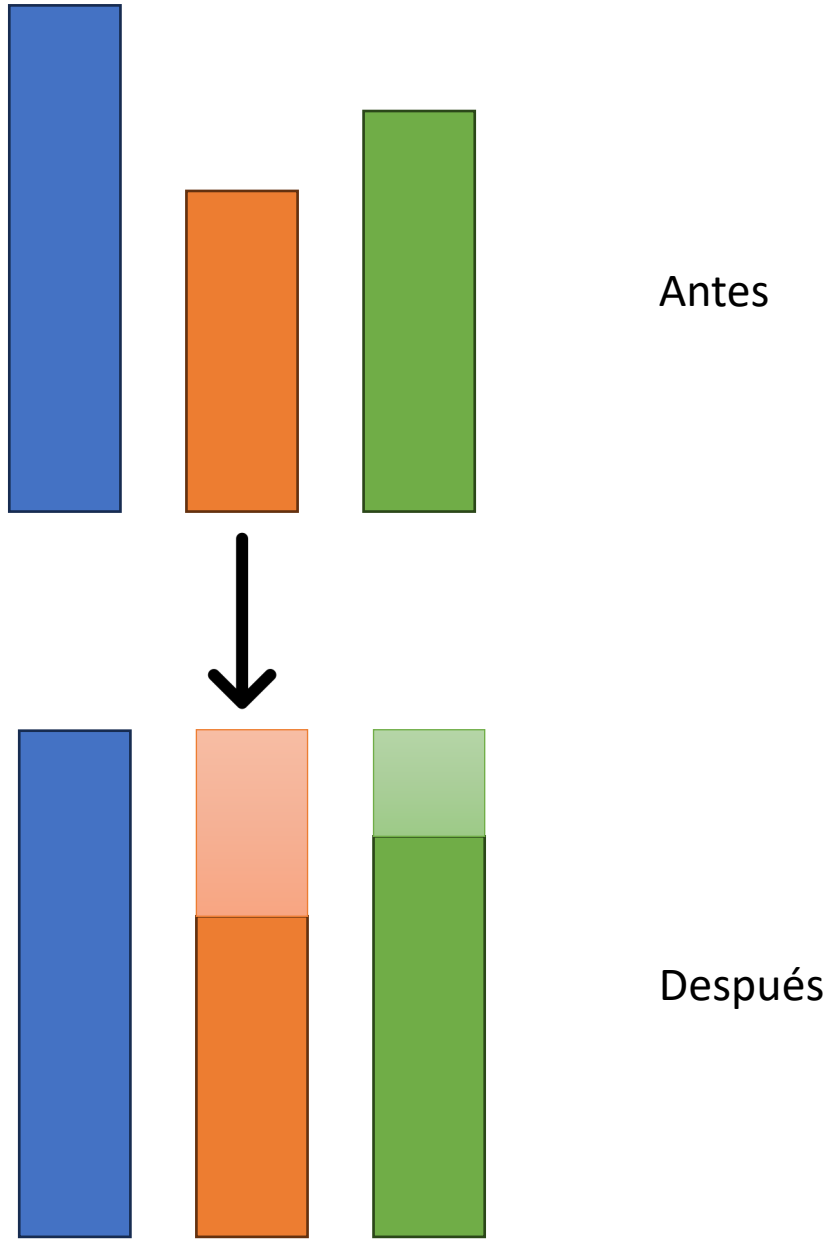
- Ahora, vamos a incrementar las clases menos representadas con datos sintéticos.
- Suele ser acompañada por downsampling.
- El problema es generar nuevos datos. Para datos estructurados suele ser más directo (SMOTE).
- Para datos como texto, imágenes o series de tiempo suele ser más laborioso.



Patrón 7: Reequilibrar

Upsampling

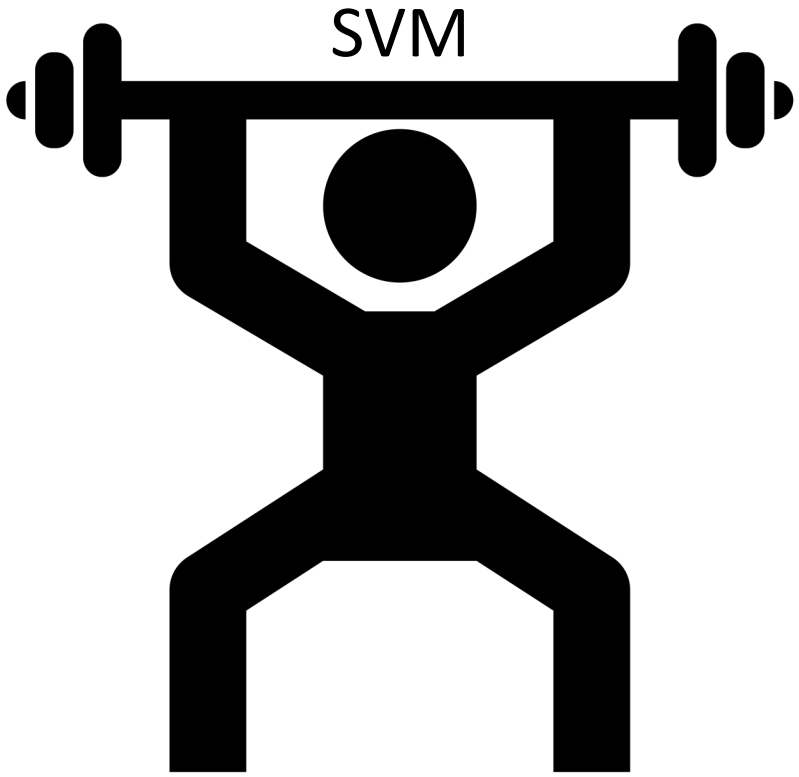
- Para texto pueden cambiar ciertas palabras (sustantivos, adjetivos) por sinónimos.
- Para imágenes es suficiente aplicar transformaciones como rotación, traslación, deformaciones, etc.
- Para series de tiempo, es más complejo.



Patrón 7: Reequilibrar

Entrenamiento penalizado

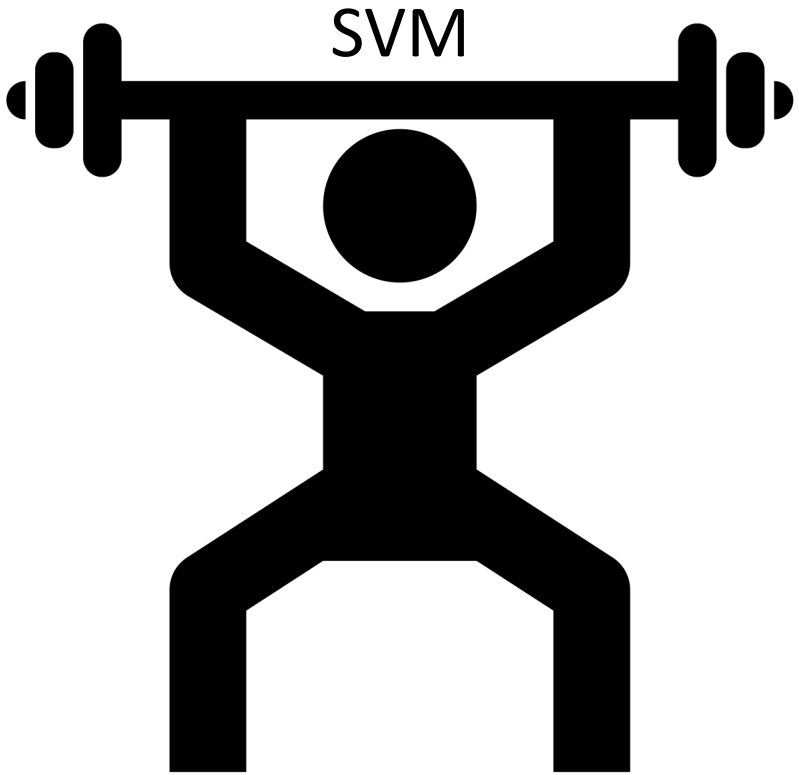
- Otra forma de atacar este problema es a nivel algoritmo.
- En este caso, vamos a modificar los pesos de las clases menos representadas para darles mayor importancia durante el entrenamiento.
- Al ser más *pesadas*, la función de pérdida les dará más importancia.



Patrón 7: Reequilibrar

Entrenamiento penalizado

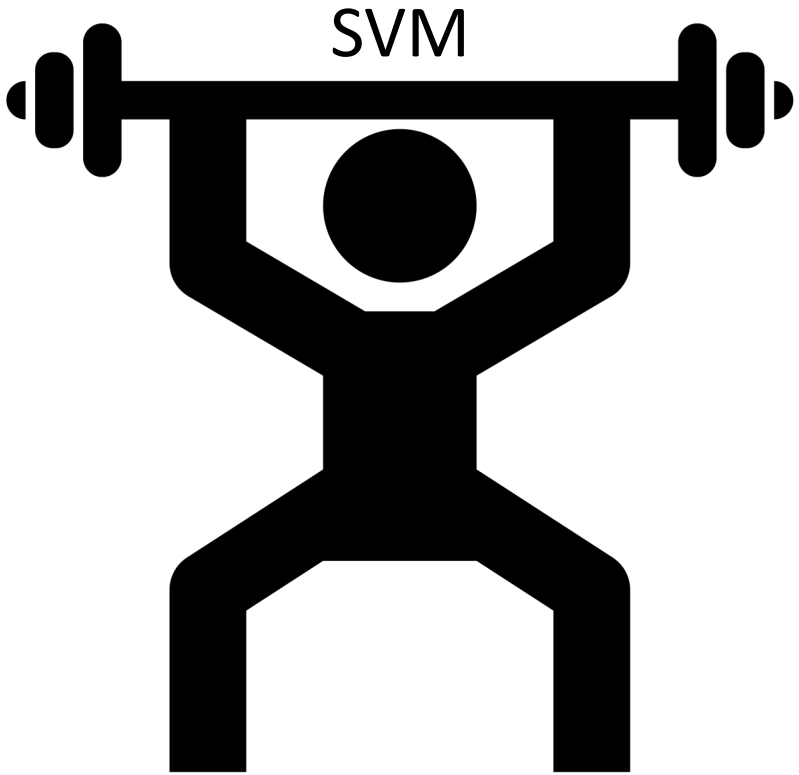
- Cada modelo tiene su forma de alterar pesos.
- Afortunadamente, librerías como Scikit-learn, Keras o Pytorch tiene formas para implementar esta idea.



Patrón 7: Reequilibrar

Entrenamiento penalizado

- Cada modelo tiene su forma de alterar pesos.
- Afortunadamente, librerías como Scikit-learn, Keras o Pytorch tiene formas para implementar esta idea.



Patrón 7: Reequilibrar

Modificar los datos

- **Siempre es mejor recopilar más datos si es posible.**
- Tener cuidad con la generación de los nuevos datos.
- No siempre es posible aplicar downsampling o upsampling.
- Al cambiar las distribuciones de los datos con downsampling o upsampling, se pierde la naturaleza de los datos.

Modificar el modelo

- Respeta la distribución natural de los datos.
- Se afecta solo el algoritmo, pero requiere proporcionar los valores de penalización.
- Otro hiperparámetro que se debe optimizar.

Patrón 7: Reequilibrar

Vamos a Google Colab para aplicar este patrón…

Conclusiones

- Analizamos diferentes vormas para representar las predicciones a través del análisis de la arquitectura de los modelos y las salidas de los modelos.
- Usando este análisis, podemos decidir qué tipo de modelo usar, o inclusive reformular esta representación y/o los algoritmos que usan para aprender.
- Aplicar estos patrones requiere una planeación con anticipación, ya que afecta los mismos datos de entrenamiento.
- Sin embargo, es posible aplicarlos después de una primera iteración de los modelos para su mejora después de un análisis de error.




Final de la presentación

¡Gracias por su atención!