

¿Cómo lidiar con los problemas desbalanceados en ML?

25 DE JUNIO, 2020

Francisco J. Llaneza

Senior Data Scientist

Deloitte Consulting – Analytics

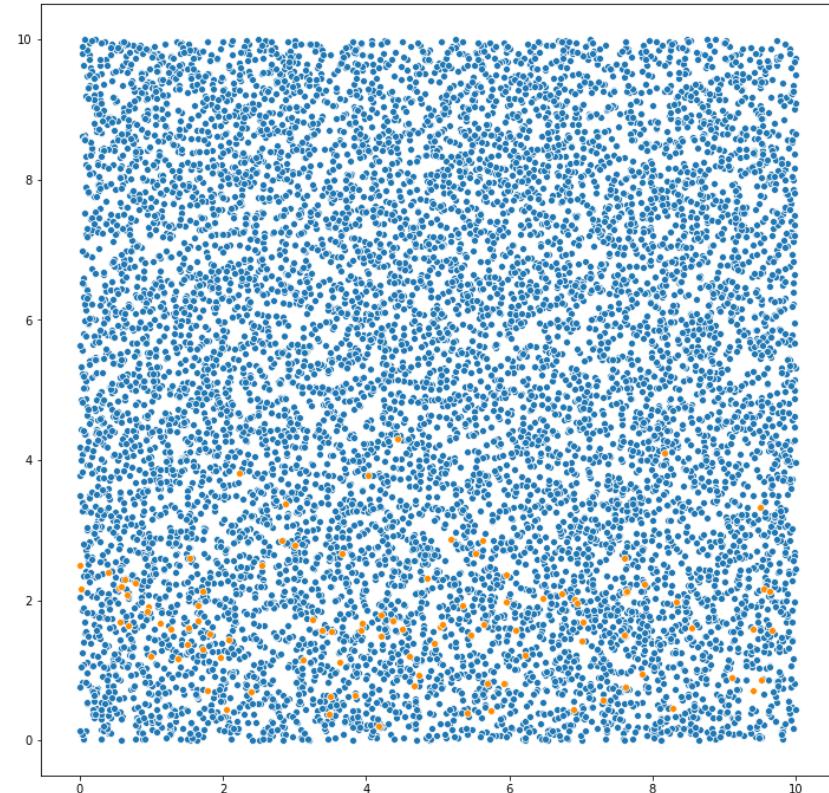
Contact: fllaneza@deloitte.es



Introducción

“Predecir lo improbable = Problema desbalanceado”

- En **Machine Learning**, existen situaciones en las cuales nos interesa especialmente predecir eventos (**clases**) “altamente improbables”.
- Este tipo de problemas se conocen como **desbalanceados**, precisamente porque los datos muestran un desequilibrio en la cantidad de ejemplos pertenecientes a cada clase
- Se suele considerar un problema como (peligrosamente) desbalanceado cuando el porcentaje de observaciones pertenecientes a la clase de interés es del **orden del 1%... ¡o menor!**



Casos de uso habituales



PROPENSIÓN DE FUGA DE
CLIENTES
(*CUSTOMER CHURN*)



DETECCIÓN DE FRAUDE



DETECCIÓN DE ANOMALÍAS



MANTENIMIENTO
PREDICTIVO

DIAGNÓSTICO MÉDICO



FILTRADO DE SPAM



PREDICCIÓN DE ABSENTISMO
LABORAL

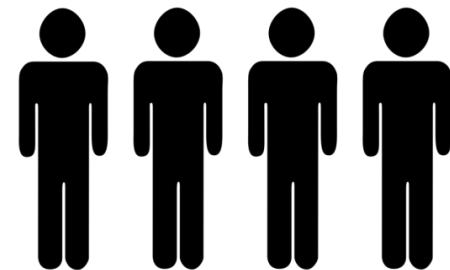


...



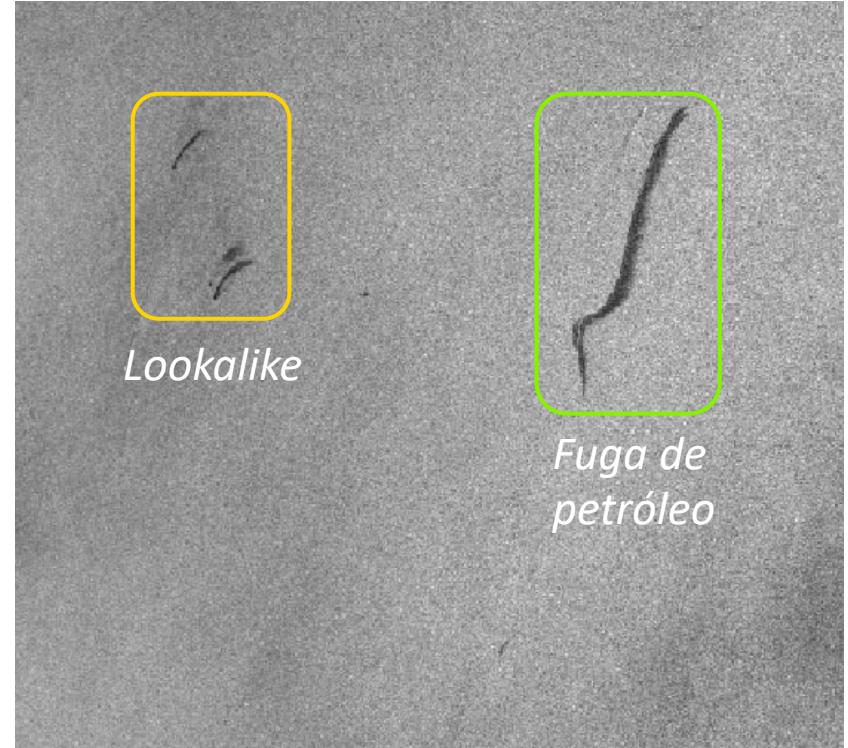
Ejemplo #1: Propensión de churn

- ¿Cómo **definir** el "abandono de clientes" en el sector bancario?
- Churn **2%** vs No Churn **98%**
- Proceso de modelización:
 - Proceso intenso de **feature engineering**
 - **Clustering** para conocer a los distintos tipos de clientes en términos de actividad dentro del banco
 - Mejor clasificador: **Gradient Boosting**
- Priorización de acciones de retención:
 - Utilizando **A/B testing**
 - Mediante un **score personalizado** teniendo en cuenta el output del modelo y el valor del cliente



Ejemplo #2: Detección de fugas de petróleo

- Escasez de imágenes con fugas de petróleo
- Desbalanceo entre casos positivos (fugas de petróleo) y casos negativos (fenómenos naturales, como lluvia o alga: *lookalikes*). 4.6% vs 95.4%
- Imágenes de **distintos grupos**, cada uno de ellos con una potencial configuración distinta del sistema de imágenes de radar
- El algoritmo únicamente filtra casos potenciales; se requiere de **conocimiento experto**



Machine Learning for the Detection of Oil Spills in Satellite Radar Images, 1998.

M. Kubat, R. Holte, S. Matwin

El gran “problema”... de los problemas desbalanceados

Pregunta...

¿Es un buen modelo?

Modelo predictivo con un alto porcentaje de acierto (accuracy)

NO necesariamente

Por ejemplo, pensemos en un modelo de diagnóstico en el campo con la siguiente matriz de confusión:

		Real		Total
Predicción	Positivo	Negativo		
	Positivo	0	0	
Negativo	1.000	99.000	100.000	

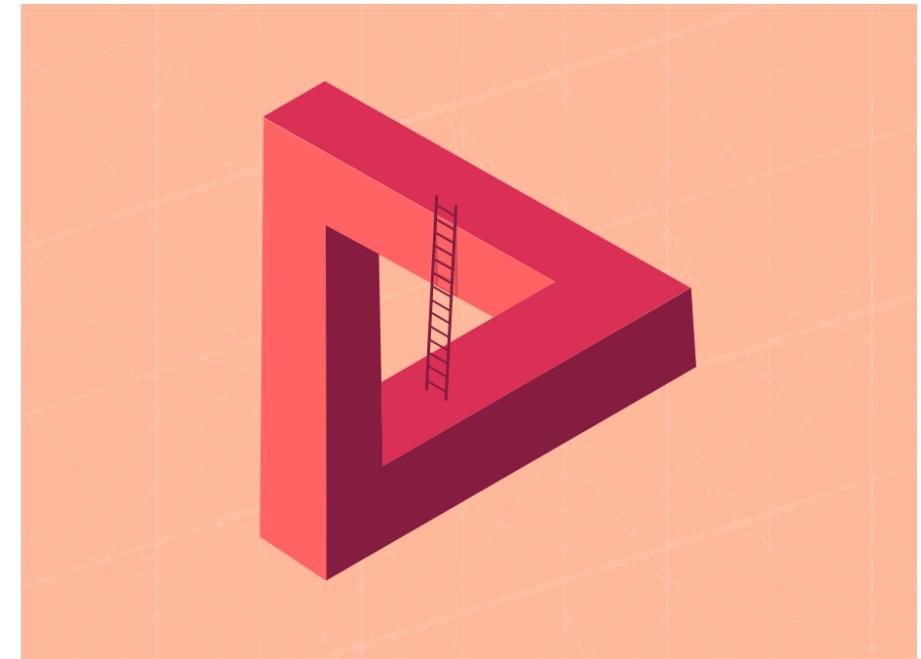


- **Acierto = $99.000 / 100.000 = 99\%$**
- **Recall = $0 / 1.000 = 0\%$**

¿Qué es y por qué ocurre esto?

“Accuracy Paradox”

- Los modelos se centran en el aprendizaje de las características de las observaciones de la(s) clase(s) mayoritaria(s), **descuidando los ejemplos de la clase minoritaria**, que es, de hecho, la de interés, y cuyas predicciones son más valiosas
- En estos caso, uno se encuentra con la paradoja de tener un modelo con un **acerto increíble**, pero completamente **inútil** para sus intereses
- Esto no es más que una consecuencia del diseño de la mayoría de algoritmos de clasificación, que se basan en la hipótesis de una **distribución equitativa** (aprox.) de las clases.



¿Qué podemos hacer para evitarlo?

Las soluciones existentes se pueden agrupar en los 3 siguientes tipos:



Métricas

- Matriz de confusión
 - Recall
 - Precision
 - F1-score
- Curvas
 - ROC
 - Precision-Recall



Datos

- Under-sampling
 - Random
 - Tomek-Links
 - Clustered Centroids
 - Edited Nearest Neighbours
 - Condensed Nearest Neighbour
- Over-sampling
 - Random
 - SMOTE
 - ADASYN

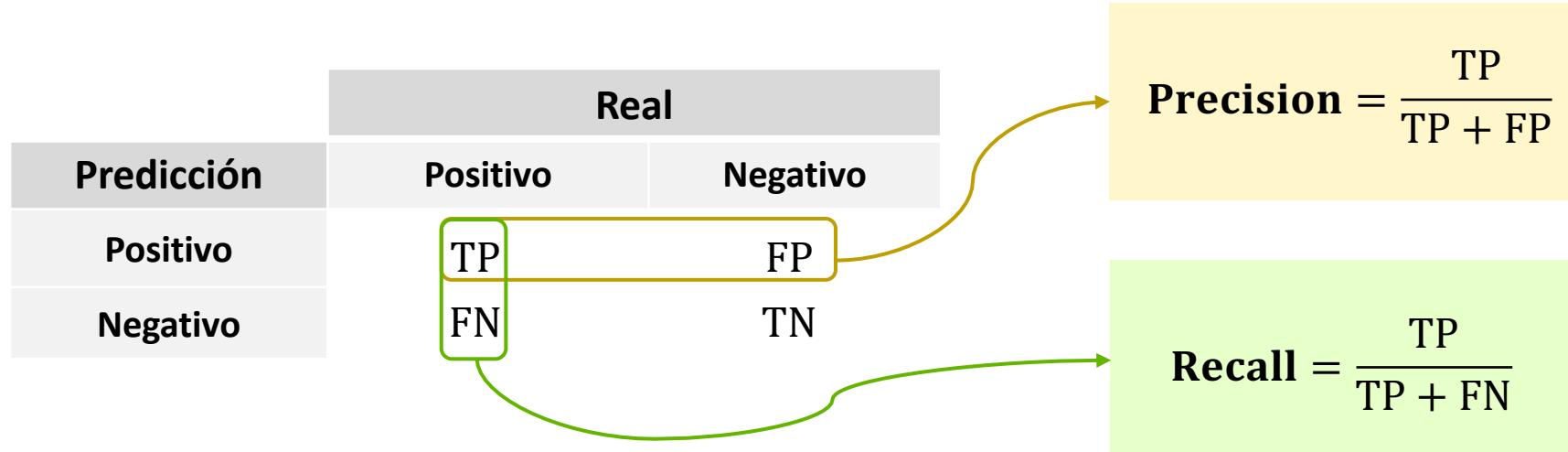


Modelos

- Class Weighting
- Boosting

Matriz de confusión | Recall & Precision

- Como hemos visto, en problemas desbalanceados no es recomendable utilizar la métrica Accuracy. En cambio, tiene más sentido medir el rendimiento del modelo de clasificación mediante las siguientes dos métricas, obtenidas también a partir de la **matriz de confusión** del modelo:
 - Recall:** *¿cuántas observaciones de la clase de interés (positivas) estoy clasificando bien?*
 - Precision:** *¿cuántas observaciones predichas como positivas, son en realidad positivas?*



Matriz de confusión | F_β score

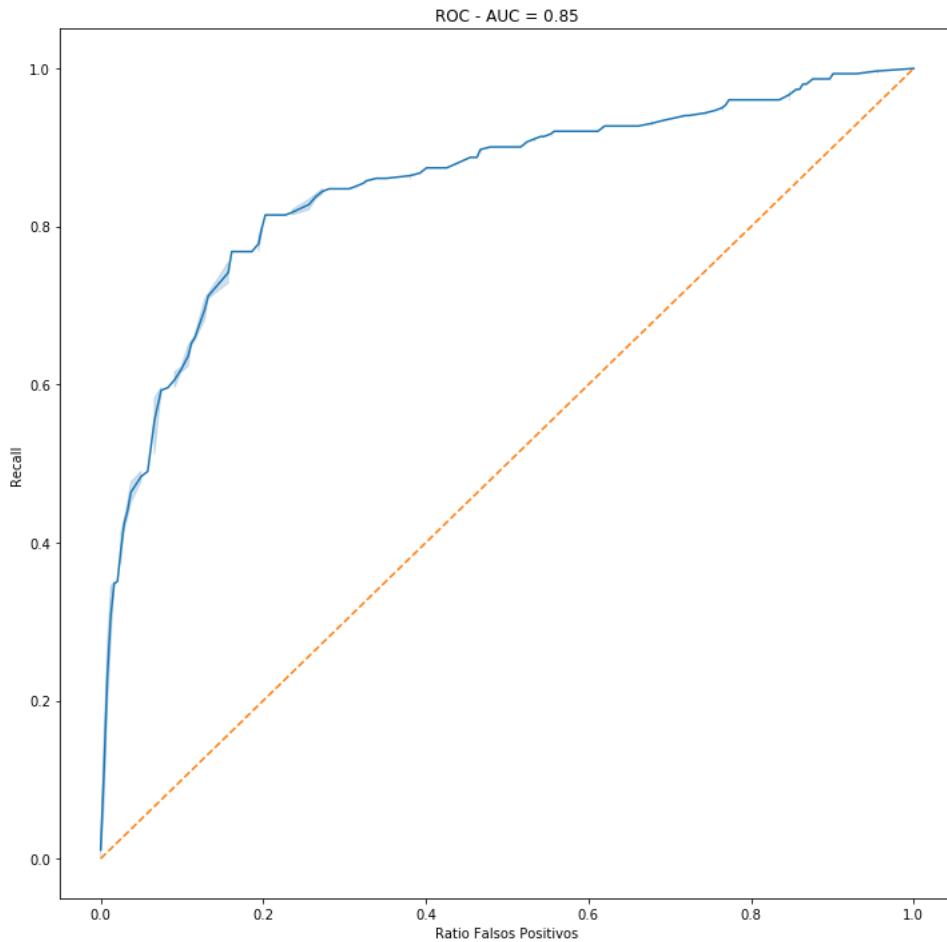
- El gran problema es que Recall y Precision tienen, en general, una **relación inversa**, por lo que mejorar una, supone empeorar la otra. Por tanto, en función del caso de uso, conviene seleccionar la una o la otra:
 - Si priorizamos capturar la **mayor cantidad** de casos positivos (aún a costa de aumentar los falsos positivos), escogeremos **Recall**
 - Si priorizamos tener una **alta confianza** a la hora de clasificar un caso como positivo (aún a costa de no capturar tantos casos positivos), escogeremos **Precision**
- Si no queremos descuidar ninguno, pero teniendo a la vez la opción de ponderar y priorizar uno más que el otro, podemos usar la siguiente métrica:

$$F_\beta \text{ score} = (1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}}$$

$\beta = 1 \rightarrow$ Misma importancia
 $\beta = 2 \rightarrow$ Prioriza Recall
 $\beta = 0.5 \rightarrow$ Prioriza Precision

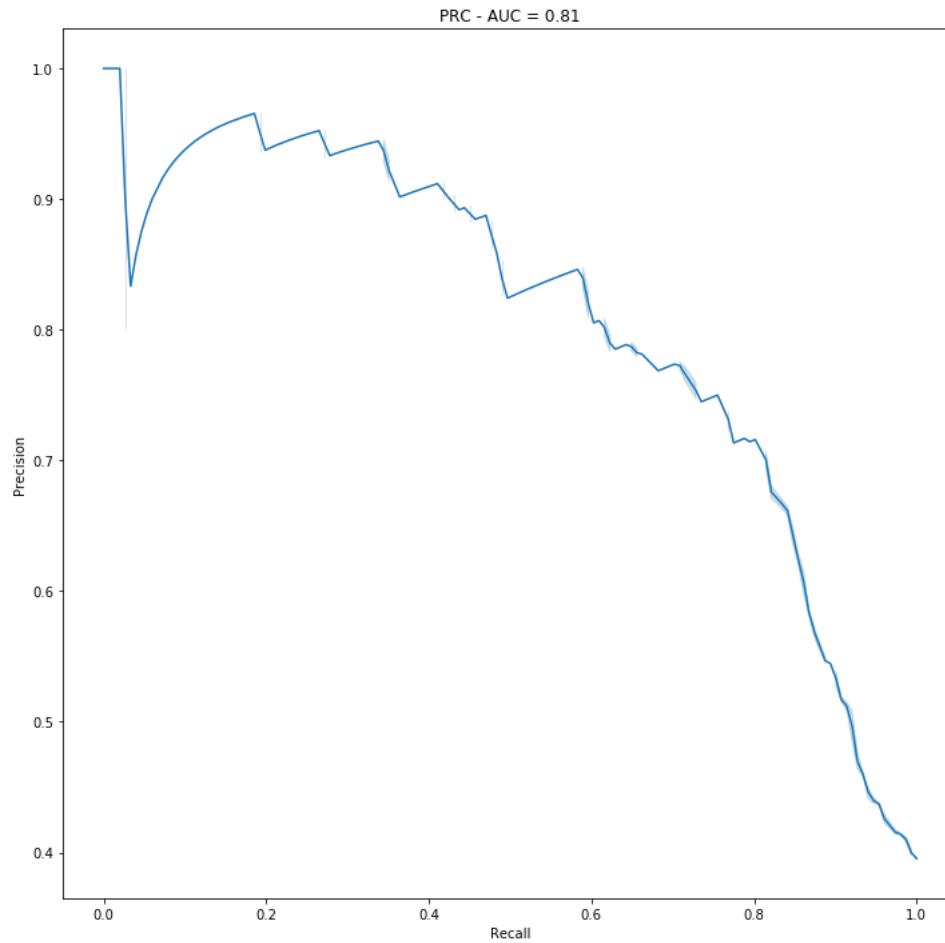
Curvas | ROC

- La curva ROC (*Receiver Operating Characteristic*) es una representación gráfica del **Recall** frente al **Ratio de Falsos Positivos** en función del umbral de probabilidad usado para la clasificación
- La información proporcionada por la curva ROC se puede condensar de forma cuantitativa en la métrica **ROC AUC**, que oscila entre 0.5 (modelo aleatorio) y 1 (modelo perfecto)
- **Principal ventaja:**
 - Al tener un baseline fijo, se puede utilizar para comparar el rendimiento de distintos modelos de forma directa
- **Principal desventaja:**
 - Puede ser demasiado optimista con problemas muy desbalanceados



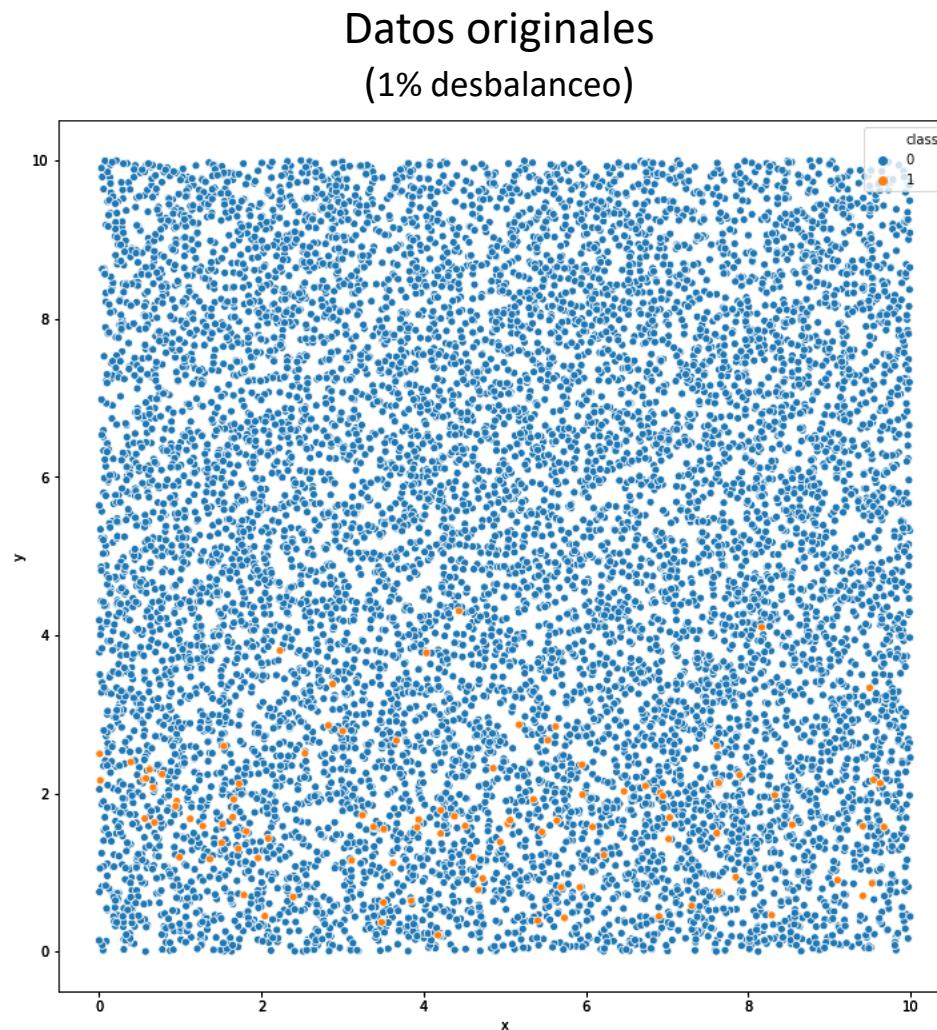
Curvas | Precision-Recall

- La curva Precision-Recall (PR) es una representación gráfica de la **Precisión** frente al **Recall** en función del umbral de probabilidad usado para la clasificación
- La información proporcionada por la curva Precision-Recall se puede condensar de forma cuantitativa en la métrica **PR AUC**
- **Principal ventaja:**
 - Es muy útil para evaluar el rendimiento de modelos en problemas severamente desbalanceados
- **Principal desventaja:**
 - El baseline depende del ratio entre clases, por lo que no se puede usar para comparar modelos construidos sobre datos con otras distribuciones



Under-sampling | Random (1/2)

- Se **reduce** el número de observaciones de la(s) clase(s) complementaria(s) a la de interés mediante un **muestreo aleatorio**
- **Principal desventaja:**
 - Se pierde información de los datos, susceptible de ser una fuente de gran valor para el modelo (“aumento de varianza del estimador”)
 - Las probabilidades de pertenencia a las clases se ven alteradas respecto a las reales (“distribución de probabilidad a posteriori deformada”)

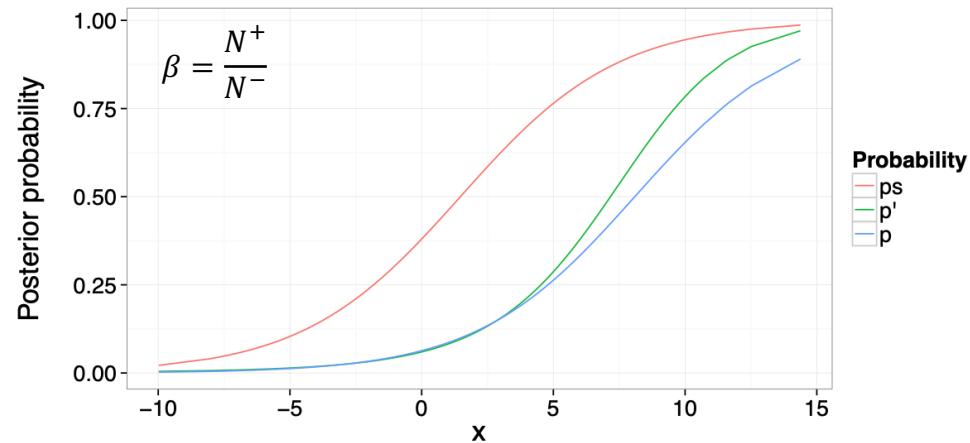


Under-sampling | Random (2/2)

- Si los **priors de las distribuciones de probabilidad** de los datos de train y test set son los mismos, se puede eliminar el sesgo introducido en la **distribución de probabilidad a posteriori** (p_s) después del under-sampling mediante la siguiente construcción:

$$p' = \frac{\beta \cdot p_s}{\beta \cdot p_s - p_s + 1}$$

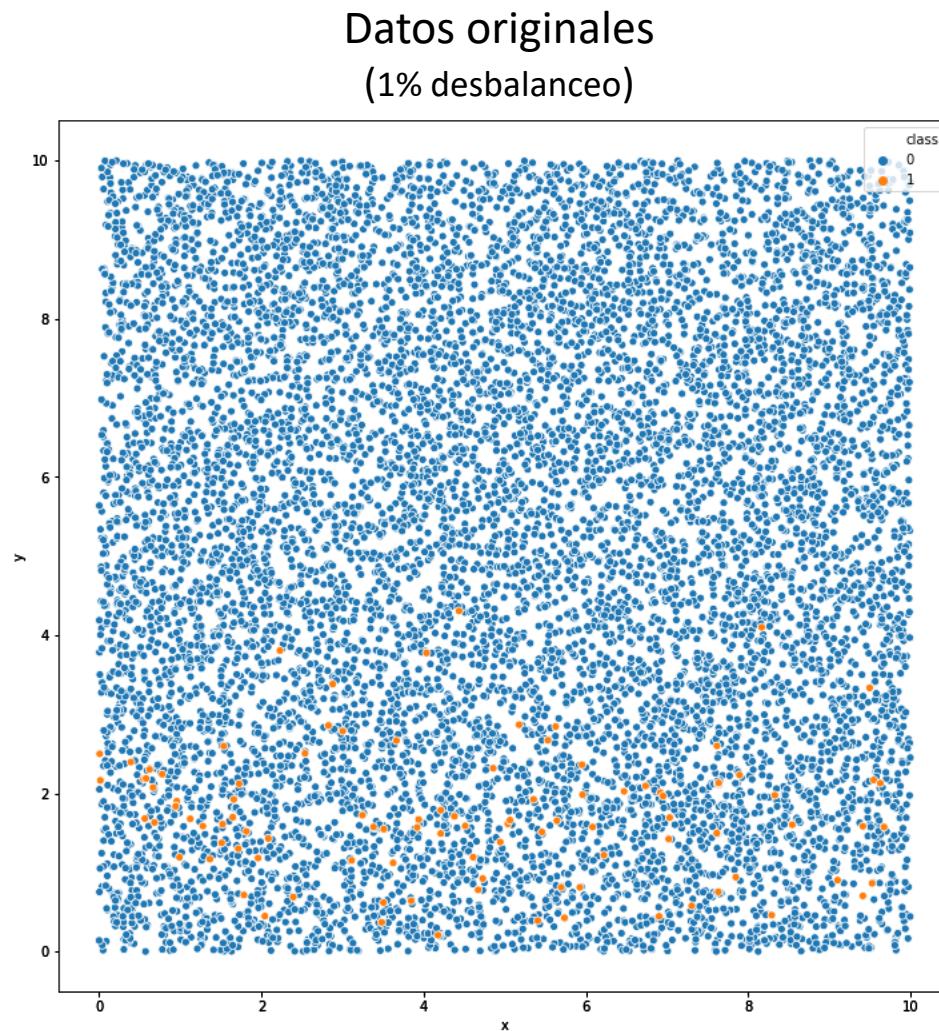
$\beta = P(\text{seleccionar} | y = 0)$



Calibrating Probability with Undersampling for Unbalanced Classification, 2015.
Dal Pozzolo, Andres. Caelen, Olivier. Johnson, Reid A. Bontempi, Gianluca

Under-sampling | Tomek's Links (1/2)

- Un **enlace de Tomek** se define como dos observaciones pertenecientes a clases distintas de tal modo que cada una es la vecina más próxima de la otra
- Eliminando estas observaciones se consigue **aumentar la separación de las clases** y, por tanto, facilitar la labor de clasificación al algoritmo. De forma adicional, si se elimina solo la observación de la clase mayoritaria, se logra un **ligero efecto de balanceo**
- **Principal desventaja:**
 - No es un método de balanceo como tal, sino de eliminación de ruido



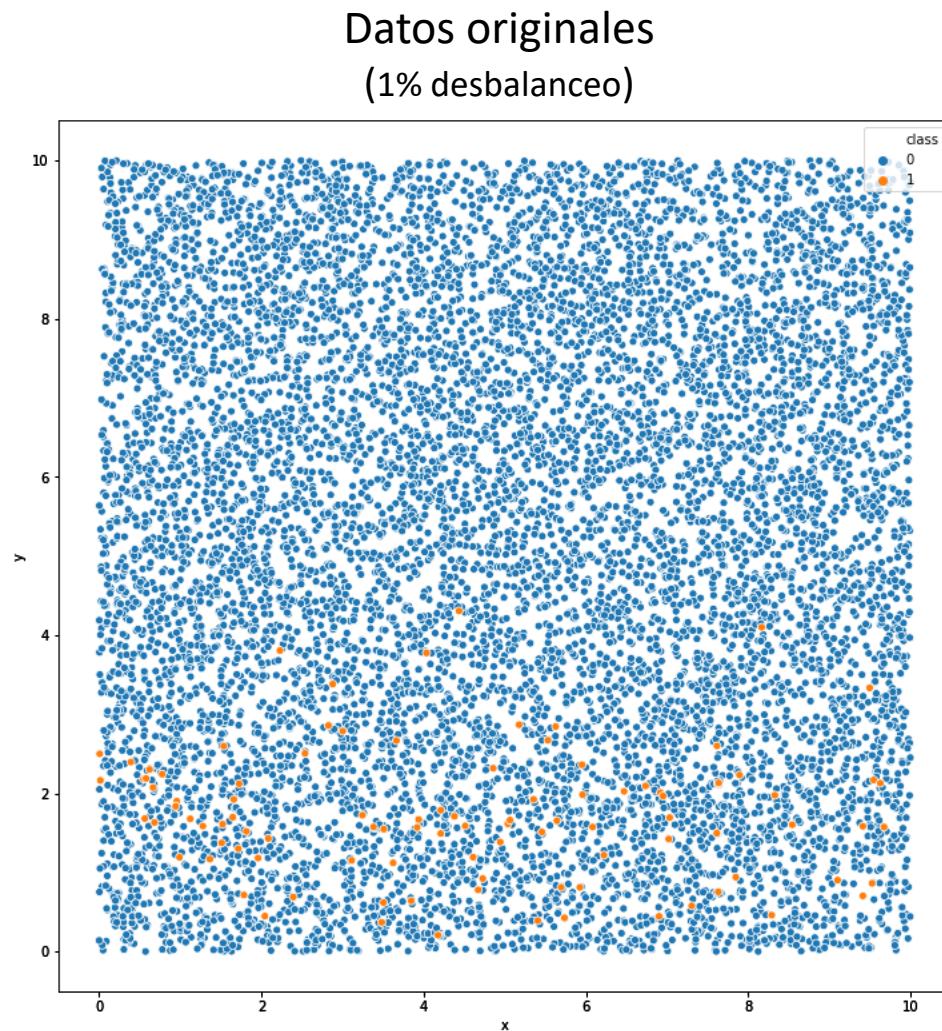
Under-sampling | Tomek's Links (2/2)



Fuente: <https://heartbeat.fritz.ai/resampling-to-properly-handle-imbalanced-datasets-in-machine-learning-64d82c16ceaa>

Under-sampling | Clustered Centroids (1/2)

- Método de sub-muestreo que funciona reemplazando grupos de observaciones de la clase mayoritaria por los **centroides de los clusters** obtenidos mediante el algoritmo K-means
- **Principales desventajas:**
 - La representatividad de la muestra obtenida se supedita a que la distribución de datos original siga los supuestos para aplicar K-means
 - No es práctico para datos con alta dimensionalidad



Under-sampling | Clustered Centroids (2/2)

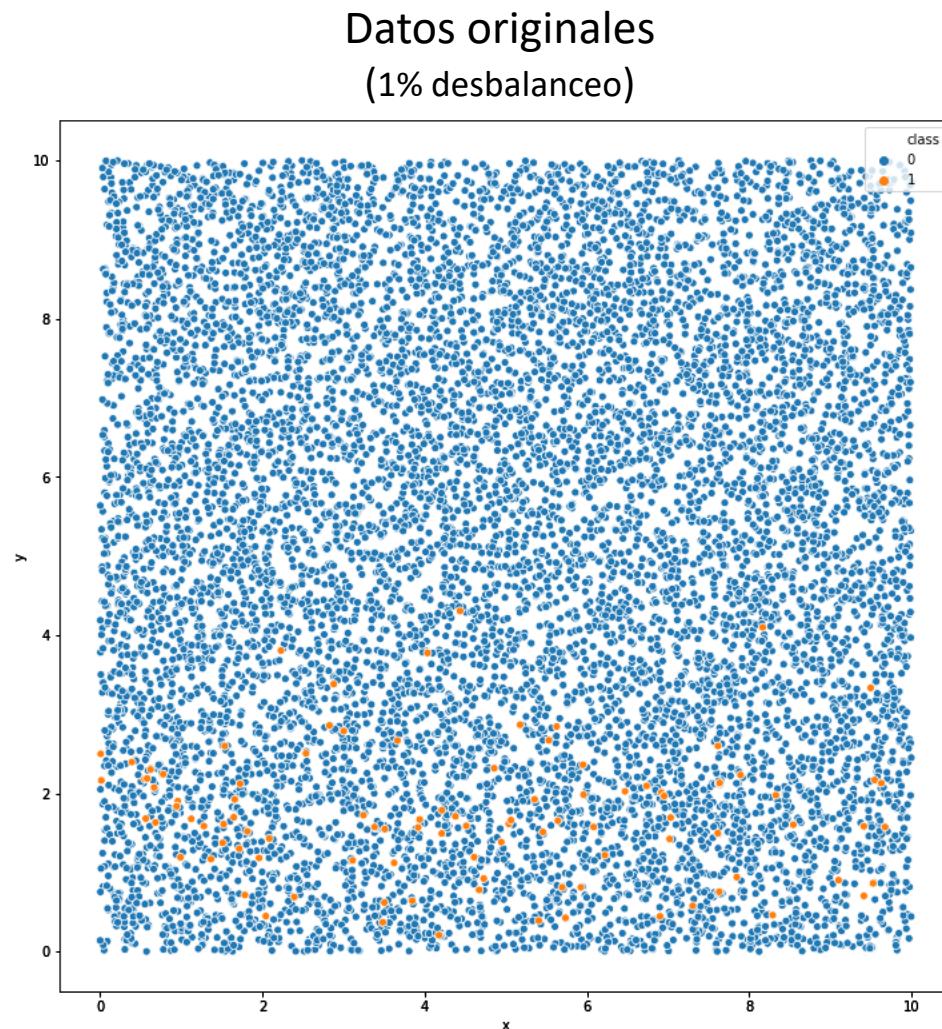


Original Dataset

Fuente: <https://heartbeat.fritz.ai/resampling-to-properly-handle-imbalanced-datasets-in-machine-learning-64d82c16ceaa>

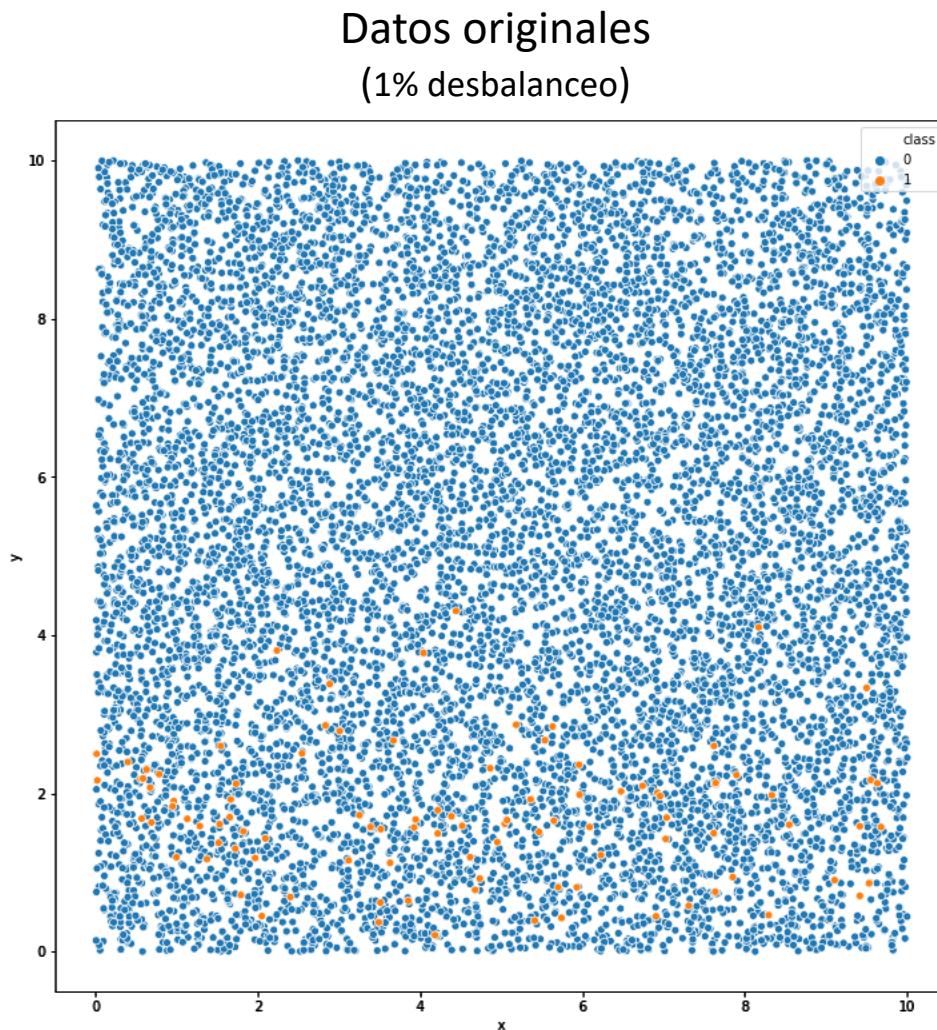
Under-sampling | Edited Nearest Neighbours

- Esta técnica permite realizar sub-muestreo eliminando aquellas observaciones de la clase mayoritaria que no se clasifican de forma correcta mediante un modelo **K-nearest neighbors** (K-NN)
- **Principales desventajas:**
 - No es práctico para datos con alta dimensionalidad



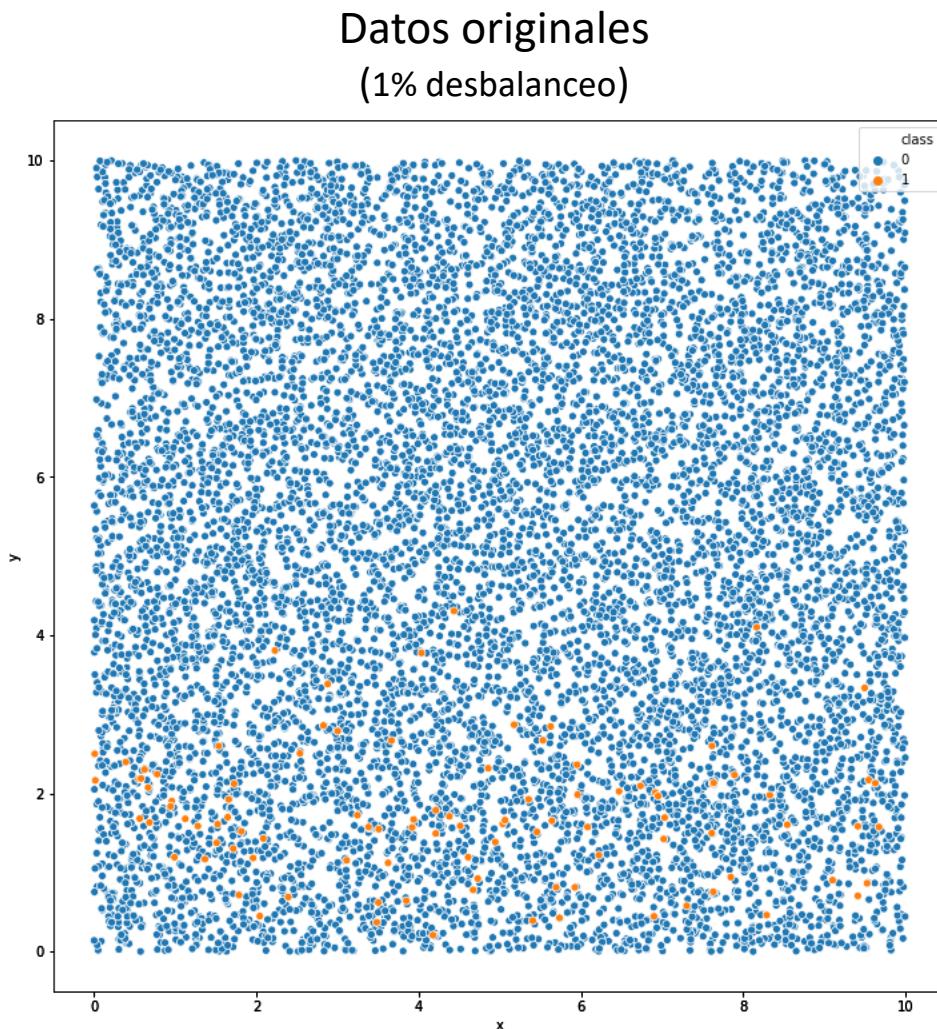
Under-sampling | Condensed Nearest Neighbour

- Método de sub-muestreo basado en el uso de un **clasificador 1-NN** (K-NN con 1-vecino más próximo) para decidir, de forma iterativa, si una observación de la clase mayoritaria debe mantenerse o no en el conjunto de datos.
- **Principales desventajas:**
 - Es muy sensible al ruido puesto que es incapaz de eliminar observaciones anómalas
 - No es práctico para datos con alta dimensionalidad



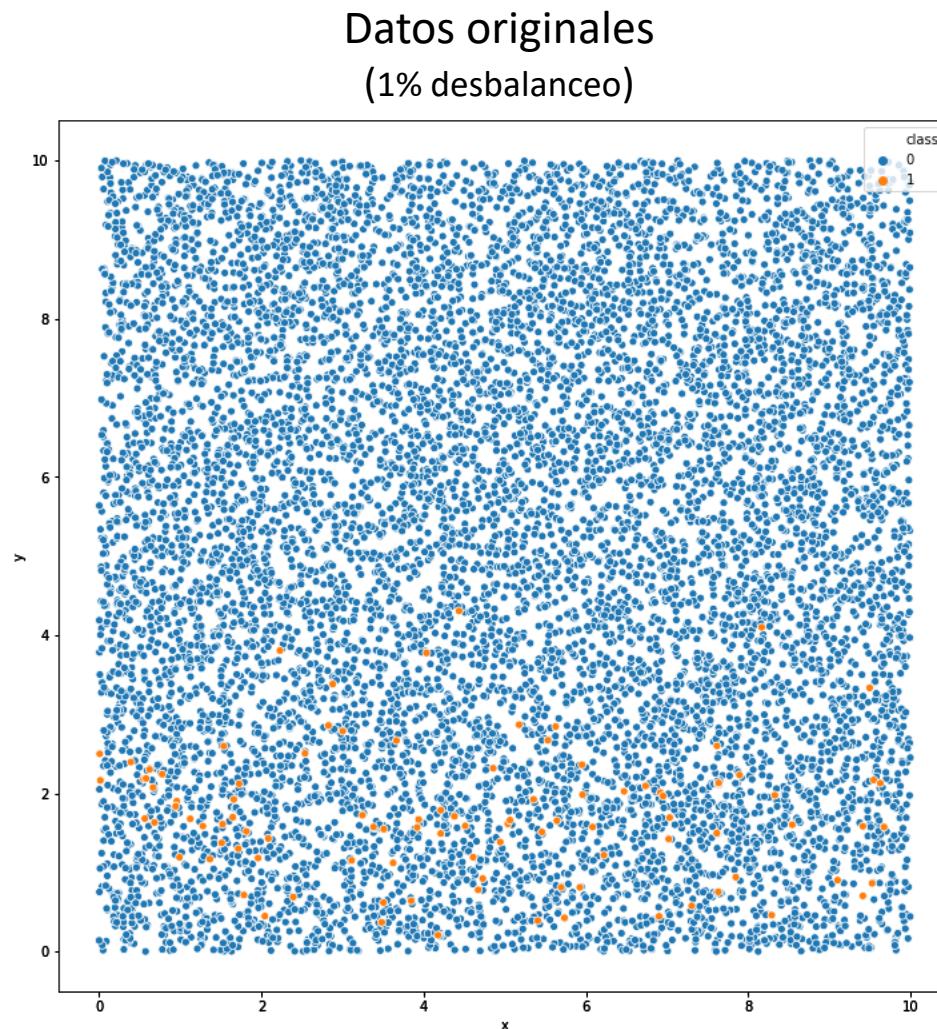
Over-sampling | Random

- Se **aumenta** el número las observaciones de la clase de interés mediante un **muestreo aleatorio y con reemplazamiento**
- **Principal desventaja:**
 - Se tiende a obtener modelos con sobreajuste (es decir, sin buena capacidad de generalización) en la clase de interés



Over-sampling | SMOTE (1/2)

- SMOTE (*Synthetic Minority Over-sampling Technique*) consiste en una técnica de sobre-muestreo basada en la generación de **nuevas observaciones sintéticas** de la clase de interés mediante combinaciones de observaciones existentes (combinaciones lineales convexas con los K-vecinos más próximos)
- **Principales desventajas:**
 - Generaliza de forma ciega el área de la clase minoritaria al no tener en cuenta la clase mayoritaria, lo que puede producir solapes entre clases y aumentar el ruido
 - No es práctico para datos con alta dimensionalidad



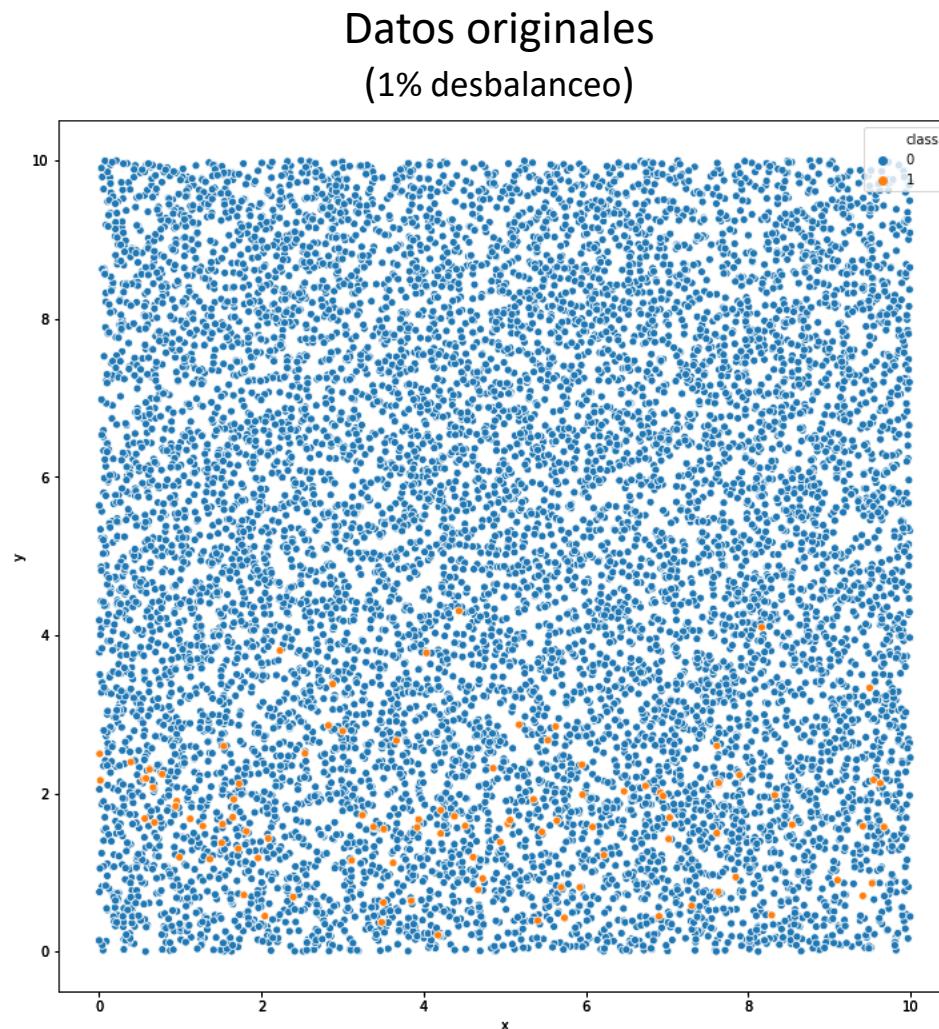
Over-sampling | SMOTE (2/2)



Fuente: <https://heartbeat.fritz.ai/resampling-to-properly-handle-imbalanced-datasets-in-machine-learning-64d82c16ceaa>

Over-sampling | ADASYN

- ADASYN (*Adaptive Synthetic*) se basa en la metodología de **SMOTE** para crear observaciones sintéticas, pero con el añadido de considerar la distribución de la **clase mayoritaria** como criterio para decidir automáticamente el número de observaciones a generar por cada observación de la clase minoritaria
- **Principales desventajas:**
 - No es práctico para datos con alta dimensionalidad



Class Weighting (1/4)

- La función de coste de los modelos de clasificación que se suele utilizar es la **cross-entropy**. Para el caso binario, se tiene la siguiente fórmula:

$$L(y_{real}, y_{pred}) = -y_{real} \log(y_{pred}) - (1 - y_{real}) \log(1 - y_{pred})$$

Clase de interés (1)

Clase complementaria (0)

- Podemos **ponderar** cada uno de los dos términos para que durante el entrenamiento del modelo se dé más importancia a una clase u otra.

$$L(y_{real}, y_{pred}) = -w_1 y_{real} \log(y_{pred}) - w_0 (1 - y_{real}) \log(1 - y_{pred})$$

Class Weighting (2/4)

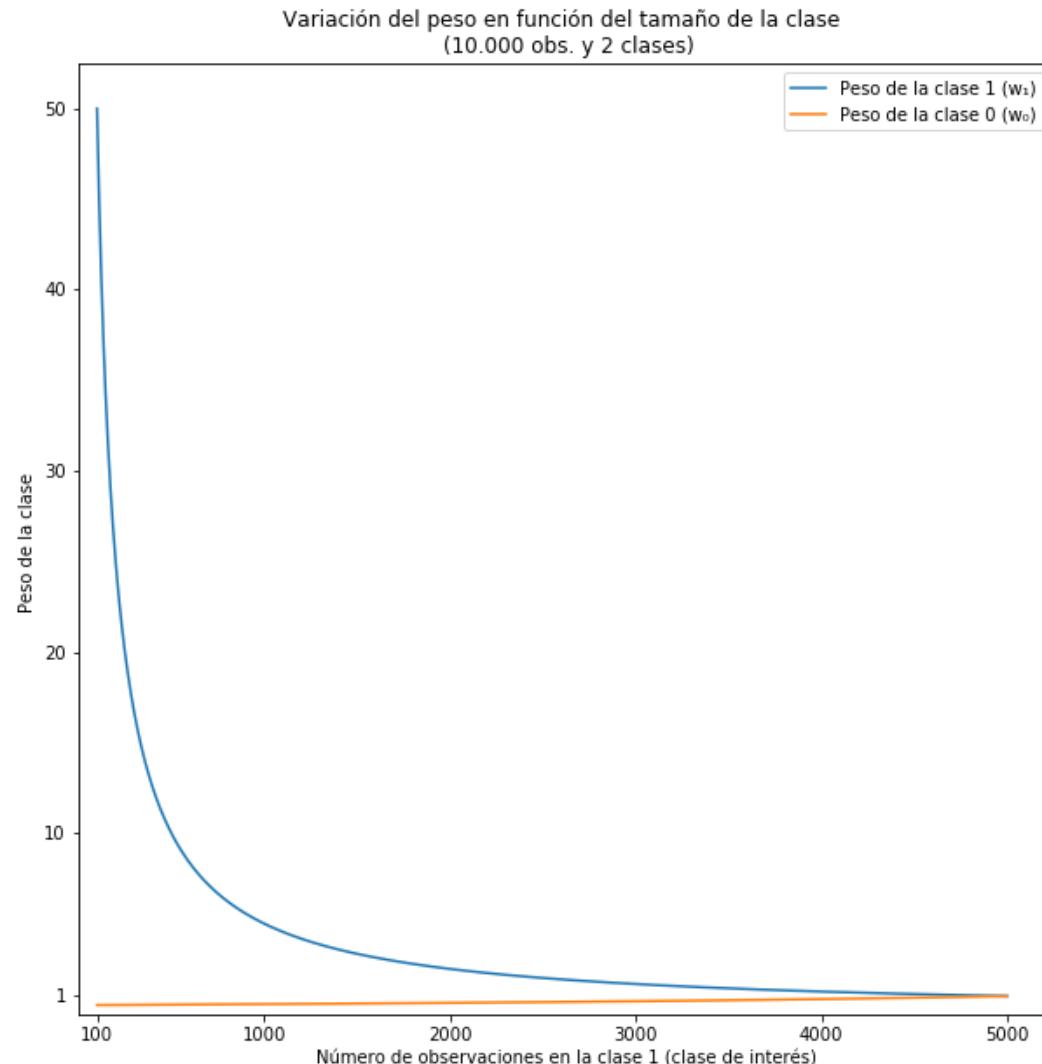
- En el caso del desbalanceo, se asigna **más peso a la clase de interés** (minoritaria) que a la clase complementaria (mayoritaria), para así balancear el aprendizaje. Por ejemplo:

$$L(y_{real}, y_{pred}) = -10y_{real} \log(y_{pred}) - 0.5(1 - y_{real}) \log(1 - y_{pred})$$

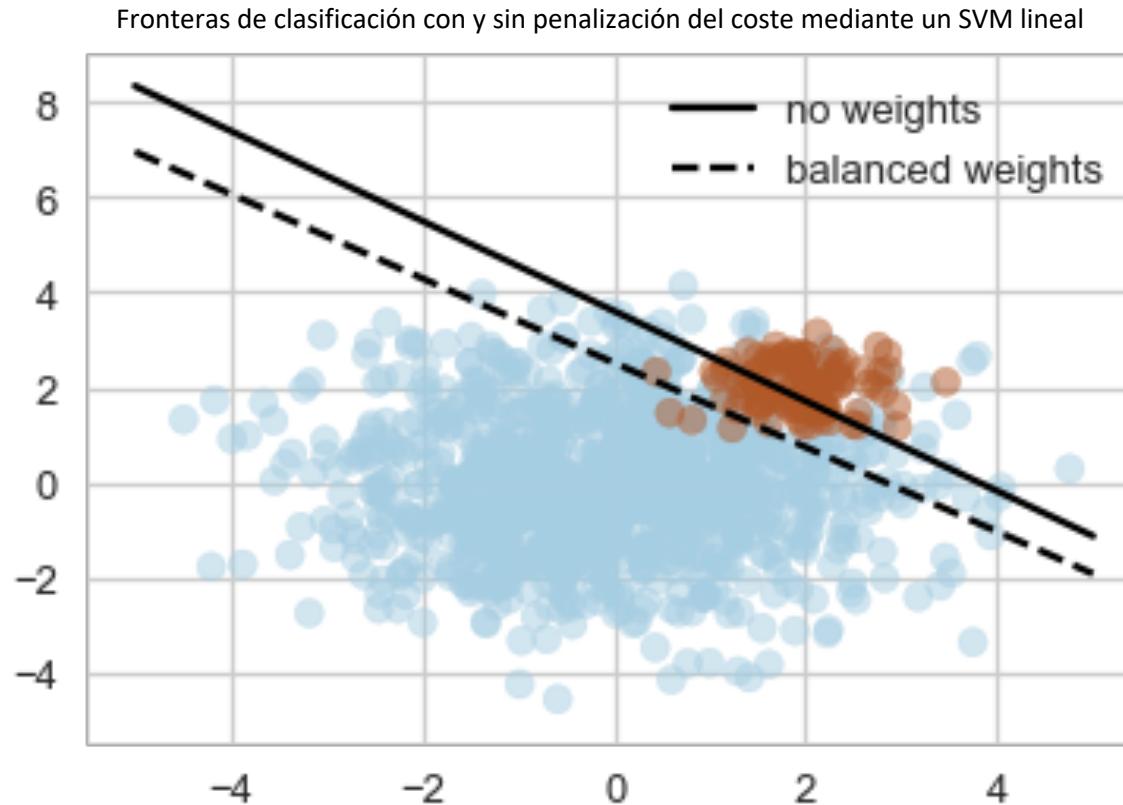
- Existe un **método heurístico** para la elección de pesos:

$$w_i = \frac{\text{Número de observaciones}}{\text{Número de clases} * \text{Número de observaciones en la clase } i}$$

Class Weighting (3/4)



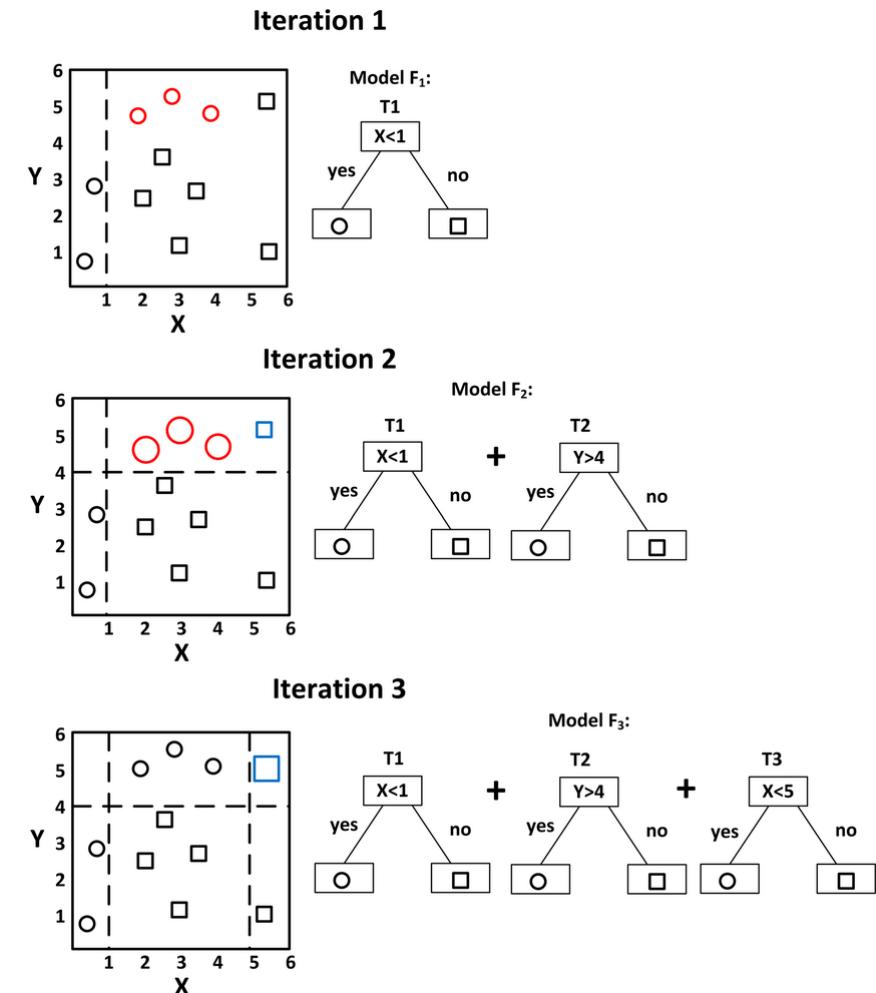
Class Weighting (4/4)



Fuente: <https://towardsdatascience.com/practical-tips-for-class-imbalance-in-binary-classification-6ee29bcdb8a7>

Boosting

- El **Boosting** es un meta-algoritmo de combinación (ensembles) de modelos, consistente en una construcción **secuencial** de estos
- La predicción final es una **suma ponderada** de las predicciones de todos los sub-modelos
- Puede **ayudar** a lidiar con el **desbalanceo** de datos porque **pesa** las observaciones en función de lo bien o mal que hayan sido clasificadas por el sub-modelo anterior
- Ejemplos: *AdaBoost, Gradient Boosting, XGBoost*
- **Principales desventajas:**
 - Puede dar como resultado un modelo con sobre-ajuste



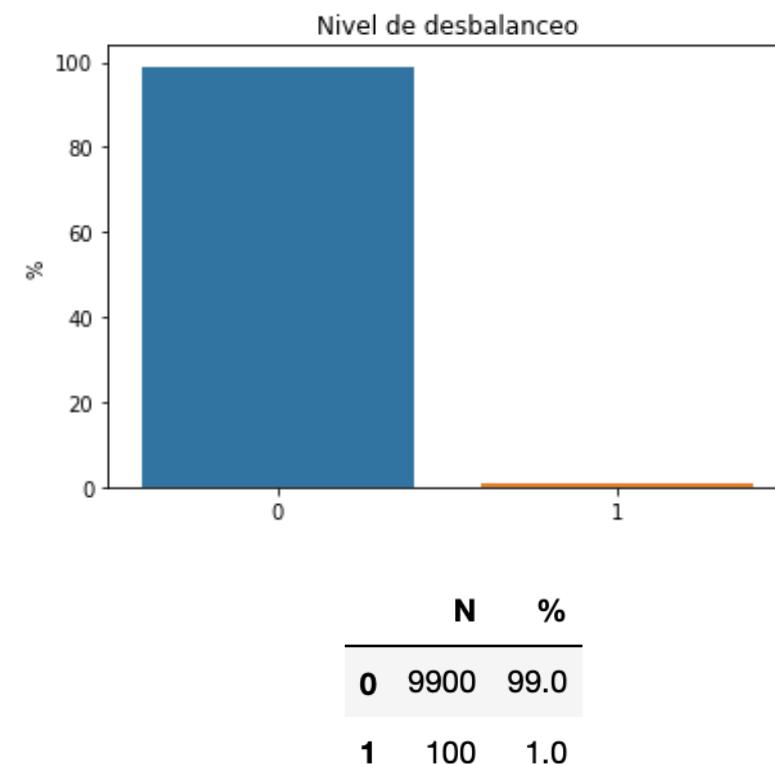
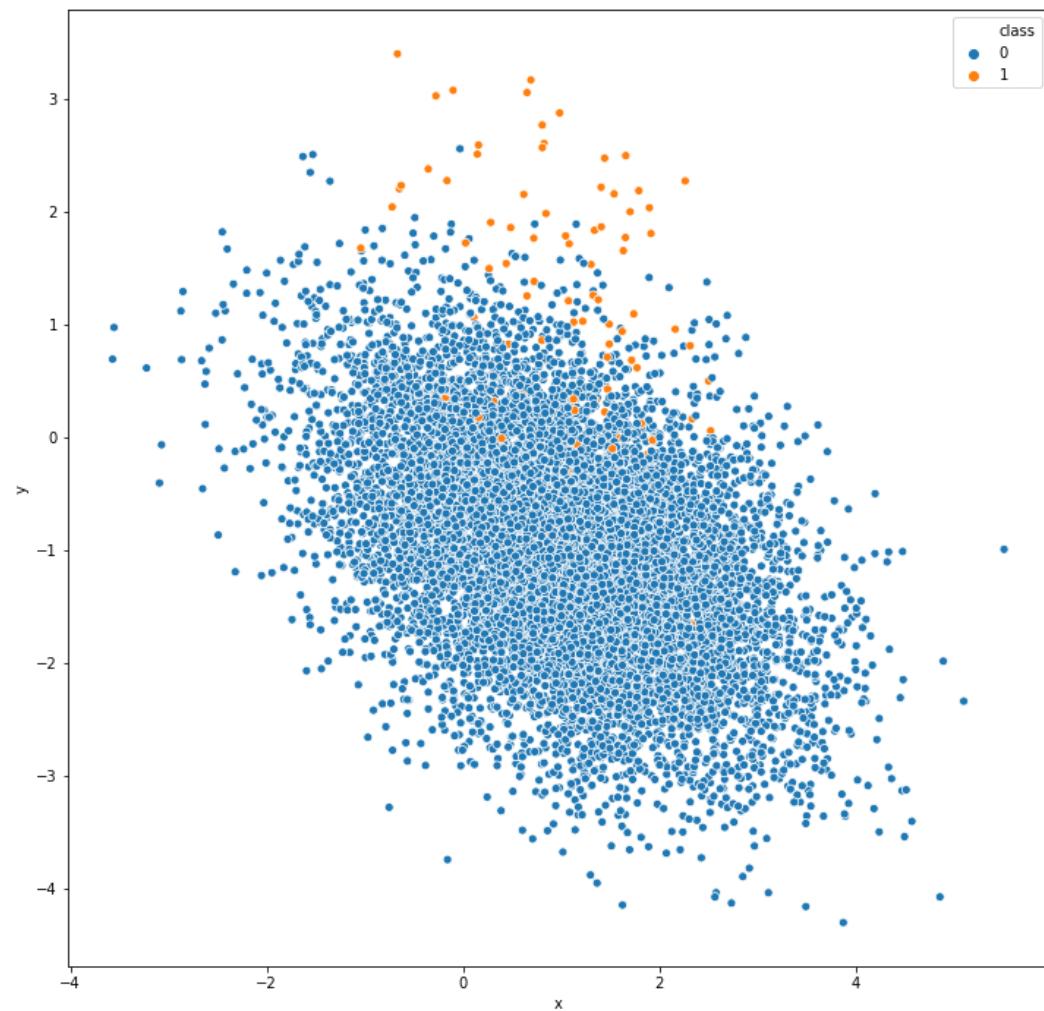
Fuente: https://www.researchgate.net/figure/A-simple-example-of-visualizing-gradient-boosting_fig5_326379229

Comparativa de técnicas (1/4)

- No se puede garantizar que ninguna técnica vaya a funcionar mejor que las demás en todas las situaciones y problemas (*"no free lunch"*)
- Sin embargo, tiene sentido hacerse una idea del grado de mejora que nos pueden llegar a proporcionar. Para ello, hemos realizado una comparativa con un dataset sintético:
 - **10.000** observaciones
 - **2** features
 - **Desbalanceo del 1%**
- La metodología que hemos seguido es la siguiente:

- ✓ **Modelo de clasificación baseline:** Árbol de decisión (hiper-parámetros por defecto)
- ✓ **Testing:** Repeated Stratified k-fold Cross-Validation (10 particiones y 5 repeticiones)
- ✓ **Métrica:** ROC AUC
- ✓ **Librerías:** scikit-learn, imbalanced-learn

Comparativa de técnicas (2/4)



Comparativa de técnicas (3/4)

```
# Generación del dataset
X_train, y_train = make_classification(n_samples=10000, n_features=2, n_redundant=0,
                                         n_clusters_per_class=1, weights=[0.99], flip_y=0)

# Modelo baseline
model = DecisionTreeClassifier()

# Estrategia de muestreo
sampling_strategy = RandomUnderSampler(sampling_strategy=0.1)

# Fases del proceso
pipeline = Pipeline([('sampling', sampling_strategy), ('classification', model)])

# Método de validación cruzada
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=5)

# Evaluación del modelo
scores = cross_validate(pipeline, X_train, y_train, scoring='roc_auc', cv=cv, n_jobs=-1)
score = np.mean(scores['test_score'])
```

Comparativa de técnicas (4/4)

		ROC AUC	Improvement
	Vanilla	66.97	
	Random Under-sampling	74.86	True
	Tomeks Links Under-sampling	68.13	True
	Clustered Centroids Under-sampling	75.56	True
	Edited Nearest Neighbours Under-sampling	70.18	True
	Condensed Nearest Neighbour Under-sampling	67.85	True
	Random Over-sampling	66.62	False
	SMOTE	75.54	True
	ADASYN	75.28	True
	Class Weighting	67.01	True
	Random Under-sampling + Random Over-sampling	72.41	True
	SMOTE + Tomeks Links Under-sampling	75.83	True
	SMOTE + Edited Nearest Neighbours Under-sampling	79.46	True
	Gradient Boosting	90.54	True

Otra consideración importante en problemas desbalanceados...

Pregunta...

¿Con qué modelo nos quedamos?

*a) Un modelo con un **80% de acierto** y una **confianza del 90%***

*b) Un modelo con un **80% de acierto** y una **confianza del 75%***

La importancia de la calibración de modelos

“¿Output del modelo = Probabilidad?”

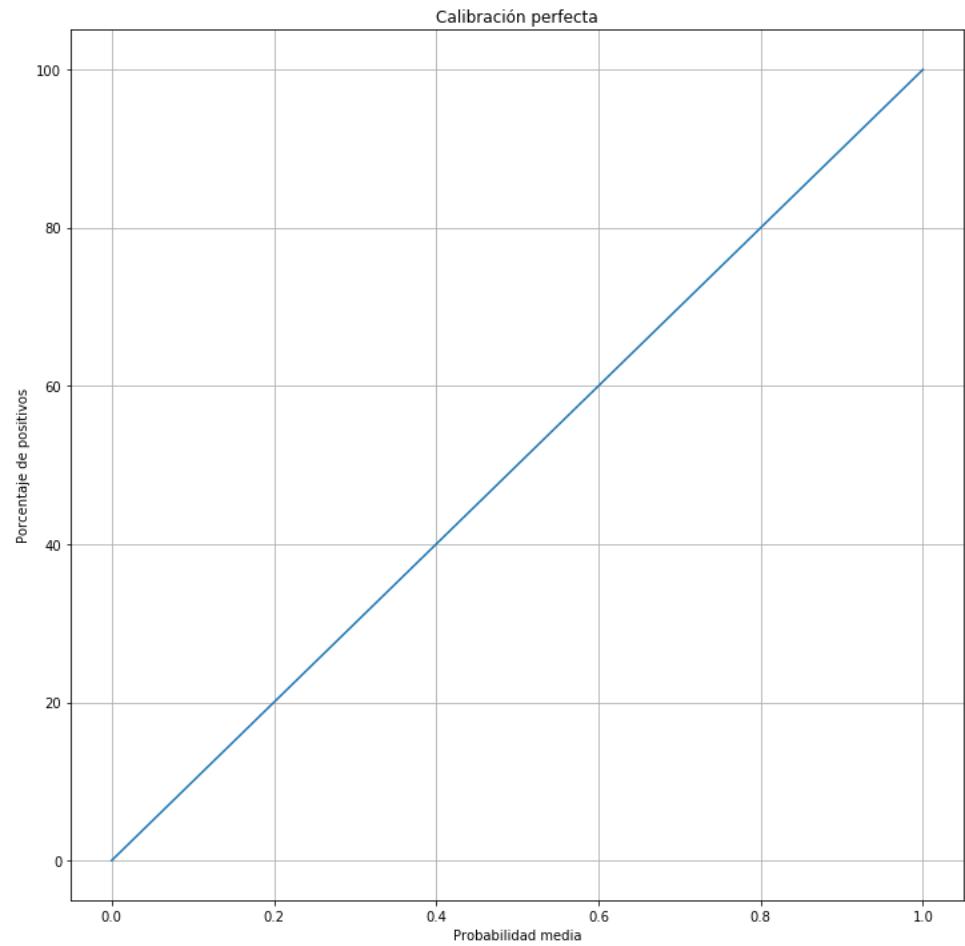
- Al realizar una predicción, un modelo puede ser “demasiado confiado” en algunos casos, y “poco confiado” en otros. Cuando esto sucede, se dice que no está **calibrado**
- En los problemas desbalanceados, esta situación se agudiza puesto que la sesgada distribución de clases puede resultar en un **sesgo** aún mayor en las **probabilidades predichas**
- Es recomendable, en general, solucionar esta situación mediante una **calibración** del modelo... y necesario, en particular, si además de la predicción de la clase nos importa la **probabilidad** asociada



Definición formal

$$P(\hat{y} = y | \hat{p} = p) = p$$
$$\forall p \in [0,1]$$

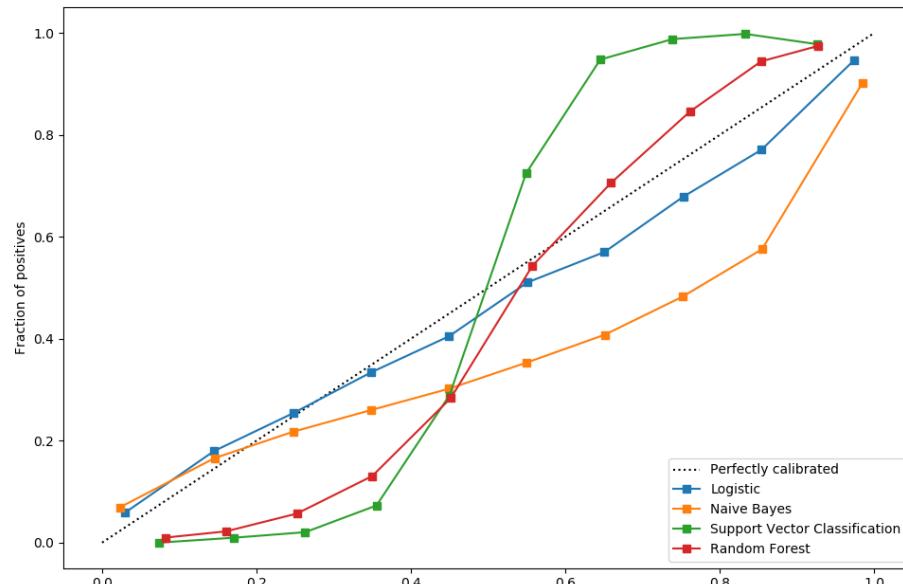
*“Un modelo bien **calibrado** clasifica las observaciones de tal modo que de las que predice un valor cercano a p , aproximadamente el $100 \cdot p\%$ pertenecen a la clase positiva”*



¿Cómo sabemos si un modelo está calibrado?

- En teoría, los modelos que funcionan bajo un **marco probabilístico** (*Regresión Logística, LDA, Naive Bayes, Red Neuronal*) devuelven **probabilidades calibradas**, mientras que los que se basan en otros métodos (*SVM, K-NN, Árboles, Ensembles*) no. Sin embargo, esto no aplica ante **problemas desbalanceados**, por lo que, en general, se conviene evaluar la calibración de un modelo mediante dos opciones (complementarias):

Curvas de calibración
(Reliability diagram)



Métricas de calibración

- Error Esperado de Calibración

$$E_{\hat{p}}[|P(\hat{y} = y | \hat{p} = p) - p|]$$

- Error Máximo de Calibración

$$\max_p |P(\hat{y} = y | \hat{p} = p) - p|$$

Métodos de calibración

- Las modelos se pueden calibrar **re-escalando las probabilidades predichas** para que se ajusten a la distribución observada en los datos
- En la práctica, esto se realiza construyendo un **modelo** capaz de tomar como input las probabilidades predichas (por el modelo de clasificación original) y devolver como output las **probabilidades esperadas**.
- En función del modelo usado para re-escalar, existen dos alternativas:

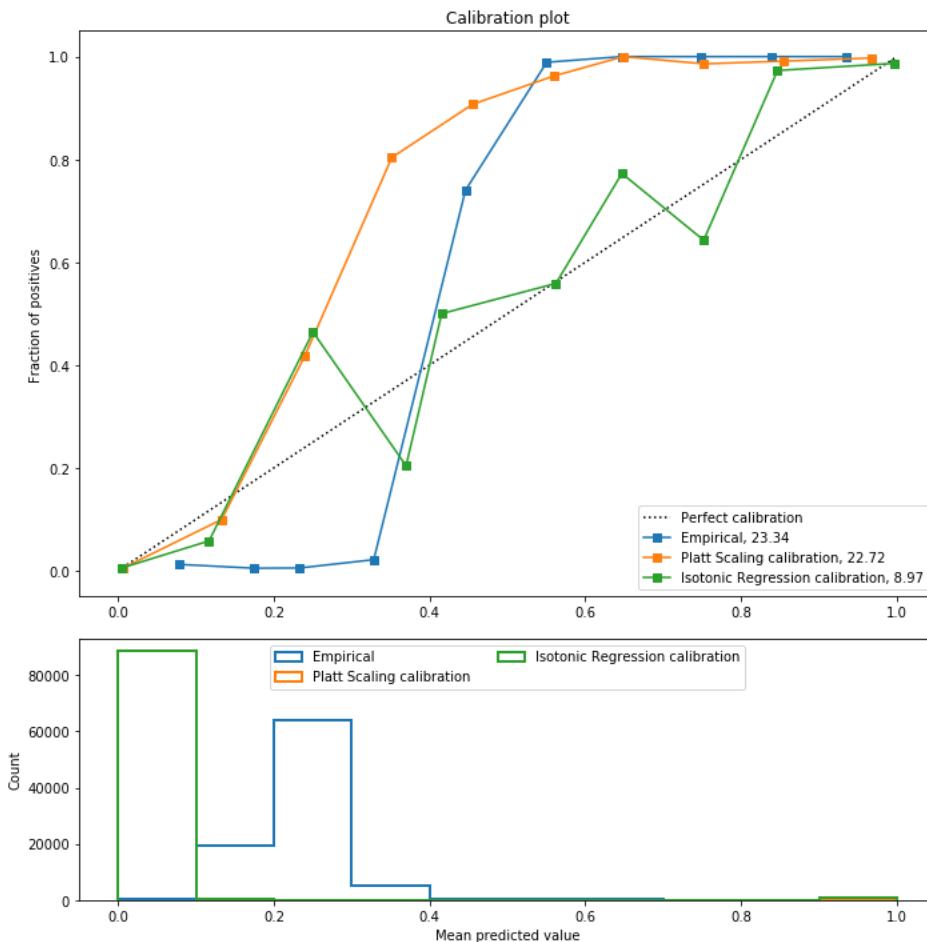
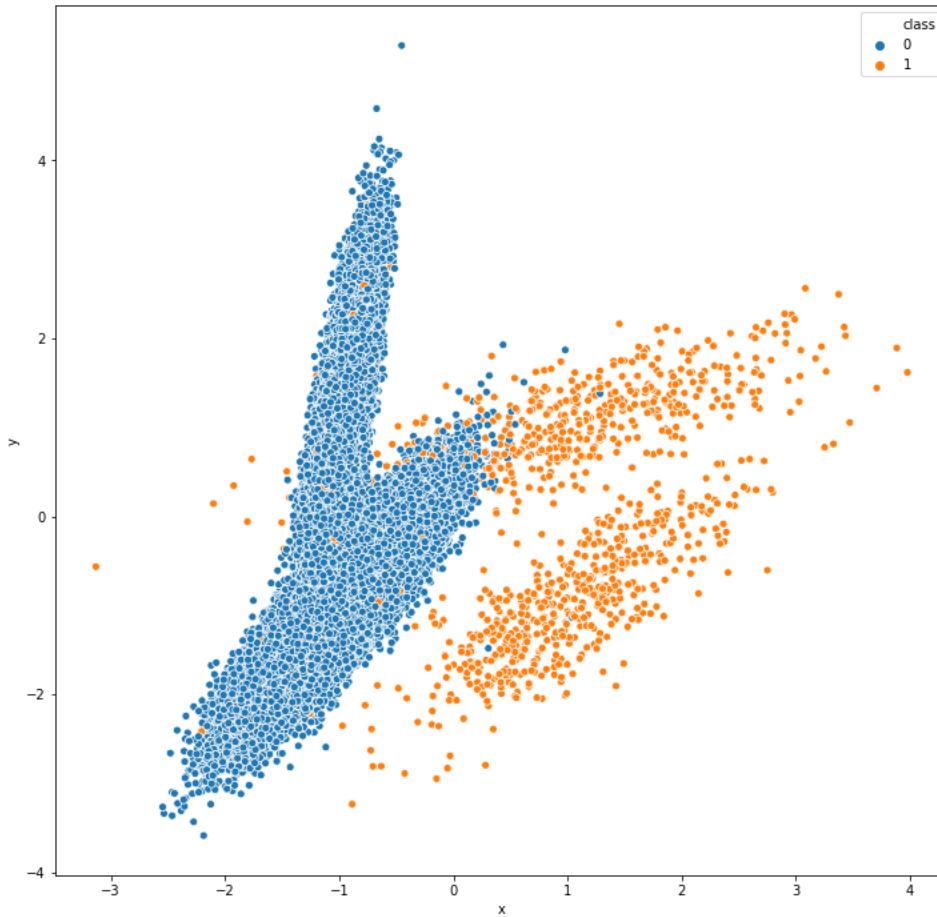
Platt Scaling

- Se utiliza una Regresión Logística
- Desarrollado para ajustar las probabilidades de un modelo SVM
- Efectivo cuando la forma de la curva de calibración tiene forma sigmoidal

Isotonic Regression

- Utiliza un modelo de regresión de mínimos cuadrados ponderados
- Es más potente y versátil
- Requiere más datos de entrenamiento
- Puede corregir cualquier distorsión monotónica

Ejemplo de calibración con SVM



Resumen

- Los problemas de datos desbalanceados surgen cuando modelizamos eventos con frecuencias muy desequilibradas
- Tenemos que utilizar métricas y criterios adecuados que nos permitan evitar conclusiones erróneas
- Existen múltiples métodos de re-muestreo y generación de observaciones para balancear las clases
- Ciertas modificaciones a nivel de modelo también han demostrado ser efectivas
- Los problemas desbalanceados ocasionan modelos no calibrados, lo cual nos afecta especialmente si nos importan las probabilidades
- Existen técnicas que permiten calibrar los modelos



¿Preguntas?



Gracias

Francisco J. Llaneza

Senior Data Scientist

Deloitte Consulting – Analytics

Contact: fllaneza@deloitte.es