

PORTUGUESE BANK MARKETING MACHINE LEARNING PROJECT

Rocío Elgorriaga

Tatiana Ursul

Julián Madoz

Leandro Bogliani



02/01/2022

CODERHOUSE

2. ÍNDICE

1. **Portada.**
2. **Tabla de contenidos / Índice.**
3. **Descripción del caso de negocio:** dedicación y problema abordado.
4. **Tabla de versionado.**
5. **Objetivos del modelo.**
6. **Link al dataset.**
7. **Variables a trabajar.**
8. **Descripción de los datos desde la perspectiva del Negocio:** Descripción de la base de datos usada en la que se explique: la temática, las variables que se seleccionaron y la segmentación de los registros, si aplica. Incluir además todas las decisiones que se tomaron para llegar al dataset que servirá de input para el algoritmo.
9. **Hallazgos encontrados por el EDA:** análisis univariado, bivariado y multivariado de las variables elegidas.
10. **Tratamiento dataset desbalanceado**
11. **Algoritmo Elegido.**
12. **Métricas de Desempeño del Modelo.**
13. **Iteraciones de Optimización.**
14. **Métricas finales del Modelo Optimizado.**
15. **Futuras líneas:** breve descripción de las posibles iniciativas que se pueden llevar a cabo para complementar el proyecto.
16. **Conclusiones:** contribuciones del modelo a los objetivos propuestos.



3. DESCRIPCIÓN DEL CASO DE NEGOCIO: DEDICACIÓN Y PROBLEMA ABORDADO.

Los depósitos a plazo fijo son una fuente importante de ingresos para un banco. Un depósito a plazo fijo es una inversión en efectivo que se mantiene en una institución financiera. Su dinero se invierte por una tasa de interés acordada durante una cantidad fija de tiempo o plazo. El banco tiene varios planes de extensión para vender depósitos a plazo a sus clientes, como marketing por correo electrónico, publicidad, marketing telefónico y marketing digital. Las campañas de marketing telefónico siguen siendo una de las formas más efectivas de llegar a las personas. Sin embargo, requieren una gran inversión ya que se contratan grandes centros de llamadas para ejecutar estas campañas. Por lo tanto, es crucial identificar de antemano a los clientes con más probabilidades de realizar una conversión para que puedan ser dirigidos específicamente a través de una llamada. Los datos están relacionados con campañas de marketing directo (llamadas telefónicas) de una institución bancaria portuguesa. El objetivo de la clasificación es predecir si el cliente se suscribirá o no a un depósito a plazo fijo (variable y).

Los datos están relacionados con las campañas de marketing directo de una institución bancaria portuguesa. Las campañas de marketing se basaron en llamadas telefónicas. A menudo, se requería más de un contacto con el mismo cliente, para poder acceder si el producto (depósito a plazo fijo bancario) estaría ('sí') o no ('no') suscrito por el cliente o no.



4. TABLA DE VERSIONADO.

Link a versiones anteriores del dataset entregados en las entregas intermedias:

1. Primera versión: [link](#)
2. Segunda versión: [link](#)
3. Tercera versión: [link](#)



5. OBJETIVOS DEL MODELO

Con el actual trabajo buscamos, en un primer lugar, realizar un análisis exploratorio de la base de datos intentando comprender el perfil que tienen los clientes que allí se encuentran registrados. De este modo -y teniendo en cuenta variables relacionadas a tanto a su contexto socio-económico, su relación con el propio banco, como con las últimas campañas de marketing que el mismo llevó a cabo-, poder predecir si los clientes, en base a sus datos históricos, se suscribieron a un depósito a plazo dentro de esta entidad bancaria o no. En este caso poseemos la variable output o 'y' por lo que se realizará un modelo de machine learning de aprendizaje supervisado. En específico, estamos trabajando un problema de clasificación donde utilizaremos diferentes algoritmos de ML para poder abordarlo.



6. LINK AL DATASET

El dataset se puede encontrar en el siguiente enlace en la página web Kaggle:

<https://www.kaggle.com/janiobachmann/bank-marketing-dataset>

Dataset

Bank Marketing Dataset

Predicting Term Deposit Suscriptions

Janio Martínez Bachmann

• updated 4 years ago (Version 1)

Data

Tasks

Code (99)

Discussion (3)

Activity

Metadata

Download (919 kB)

New Notebook

Data Explorer

918.96 kB

bank.csv

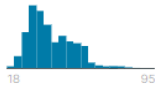
< bank.csv (918.96 kB)

Detail Compact Column

10 of 17 columns

About this file

This is the classic marketing bank dataset uploaded originally in the UCI Machine Learning Repository. The dataset gives you information about a marketing campaign of a financial institution in which you will have to analyze in order to find ways to look for future strategies in order to improve future marketing campaigns for the bank.

#	age	job	marital	education	default	#
		management 23%	married 57%	secondary 49%	true 0 0%	
		blue-collar 17%	single 32%	tertiary 33%	false 0 0%	
		Other (6652) 60%	Other (1293) 12%	Other (1997) 18%		
59		admin.	married	secondary	no	2
56		admin.	married	secondary	no	4
41		technician	married	secondary	no	1
55		services	married	secondary	no	2
54		admin.	married	tertiary	no	1
42		management	single	tertiary	no	0
56		management	married	tertiary	no	8
60		retired	divorced	secondary	no	5
37		technician	married	secondary	no	1
28		services	single	secondary	no	5
38		admin.	single	secondary	no	1
30		blue-collar	married	secondary	no	3
29		management	married	tertiary	no	1
46		blue-collar	single	tertiary	no	4
31		technician	single	tertiary	no	7
35		management	divorced	tertiary	no	3
32		blue-collar	single	primary	no	6
49		services	married	secondary	no	-1
41		admin.	married	secondary	no	5
49		admin.	divorced	secondary	no	1

Summary

- 1 file
- 17 columns



7. VARIABLES A TRABAJAR.

Descripción de las variables:

Clients:

- *age* (numeric)
- *job*: typeofjob (categorical: 'admin.', 'bluecollar', 'entrepreneur', 'housemaid', 'management', 'retired', 'selfemployed', 'services', 'student', 'technician', 'unemployed', 'unknown')
- *marital* : marital status (categorical: 'divorced', 'married', 'single', 'unknown'; note: 'divorced' means divorced or widowed)
- *education* (categorical) : 'basic.4y', 'basic.6y', 'basic.9y', 'high.school', 'illiterate', 'professional.course', 'university.degree', 'unknown')
- *default*: has credit in default? (categorical: 'no', 'yes', 'unknown')
- *housing*: has housing loan? (categorical: 'no', 'yes', 'unknown')
- *loan*: has personal loan? (categorical: 'no', 'yes', 'unknown')

Relacionado con el último contacto que tuvo con el cliente en base a la campaña actual:

- *contact*: contact communication type (categorical: 'cellular', 'telephone')
- *month*: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')
- *day_of_week*: last contact day of the week (categorical: 'mon', 'tue', 'wed', 'thu', 'fri')
- *duration*: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.

Otros atributos:

- *campaign*: number of contacts performed during this campaign and for this client (numeric, includes last contact)
- *pdays*: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
- *previous*: number of contacts performed before this campaign and for this client (numeric)
- *poutcome*: outcome of the previous marketing campaign (categorical: 'failure', 'nonexistent', 'success')

Atributos relacionados al contexto socio económico:



- *emp.var.rate*: employment variation rate - quarterly indicator (numeric)
- *cons.price.idx*: consumer price index - monthly indicator (numeric)
- *cons.conf.idx*: consumer confidence index - monthly indicator (numeric)
- *euribor3m*: euribor 3 month rate - daily indicator (numeric)
- *nr.employed*: number of employees - quarterly indicator (numeric)

Variable output (desired target):

- *y* - has the client subscribed a term deposit? (binary: 'yes','no')



8. DESCRIPCIÓN DE LOS DATOS DESDE LA PERSPECTIVA DEL NEGOCIO: DESCRIPCIÓN DE LA BASE DE DATOS USADA EN LA QUE SE EXPLIQUE: LA TEMÁTICA, LAS VARIABLES QUE SE SELECCIONARON Y LA SEGMENTACIÓN DE LOS REGISTROS, SI APLICA. INCLUIR ADEMÁS TODAS LAS DECISIONES QUE SE TOMARON PARA LLEGAR AL DATASET QUE SERVIRÁ DE INPUT PARA EL ALGORITMO.

La base de datos utilizada es la anteriormente mencionada. En base a esto fuimos pasando algunas de las variables categóricas a numéricas para poder utilizarlas dentro de los modelos de ML. Es por eso que pasamos de trabajar sobre un dataset así:

[3]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
0	56	housemaid	married	basic4y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
3	40	admin.	married	basic6y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
...
95	45	services	married	professional.course	no	yes	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
96	42	management	married	university.degree	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
97	53	admin.	divorced	university.degree	unknown	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
98	37	technician	single	professional.course	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
99	44	blue-collar	married	basic6y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no

A este:

[21]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
0	1	3	1	0	0	0	0	1	6	1	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
1	1	7	1	3	1	0	0	1	6	1	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
2	1	7	1	3	0	2	0	1	6	1	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
3	1	0	1	1	0	0	0	1	6	1	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
4	1	7	1	3	0	0	2	1	6	1	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no

De este modo y a través del [Label Encoder](#) de Sklearn pudimos encodear las variables que necesitábamos. En este caso transformamos las siguientes columnas:

1. Job
2. Marital
3. Education
4. Default
5. Housing
6. Loan
7. Contact
8. Month
9. Day_of_week

En base a esto, las variables que se seleccionaron para ser de input de los modelos fueron



las siguientes:

```
# Features
features = ['contact', 'month', 'day_of_week', 'duration', 'cons.conf.idx', 'euribor3m',
'nr.employed', 'pdays']

# Variable a predecir
target = 'y'
```

Se terminaron eligiendo estas variables debido a que, luego de probar con otras del dataset y aplicando y graficando el [feature importance](#) de las mismas, veíamos que no eran prácticamente utilizadas por los modelos y solo significaban ruido en el output.

Por otro lado, se aplicaron diferentes técnicas para tratar el desbalanceo que poseemos en nuestros datos debido a que la variable output es 'sí' y 'no' y poseemos más cantidad de de la segunda que de la primera.



9. HALLAZGOS ENCONTRADOS POR EL EDA: ANÁLISIS UNIVARIADO, BIVARIADO Y MULTIVARIADO DE LAS VARIABLES ELEGIDAS.

Al realizar el EDA nos encontramos con diversos aspectos que llamaron nuestra atención:

1. EDAD

Las edades no significan mucho, tiene una dispersión media y no tiene sentido relacionarla con otras variables ya que no nos dará ningún insight.

Esto mismo nos mostró el EDA y el feature importante ya que esta variable nunca fue utilizada en los modelos.

2. JOBS, MARITAL Y EDUCATION

De variables cómo Trabajo, Marital y Educación, el mejor análisis que se puede realizar sobre las mismas es el conteo de cada una ya que si las relacionamos con las otras, no es concluyente el análisis. Todo este tipo de variables tiene tres categorías: sí, desconocida y no para loan, default y housing.

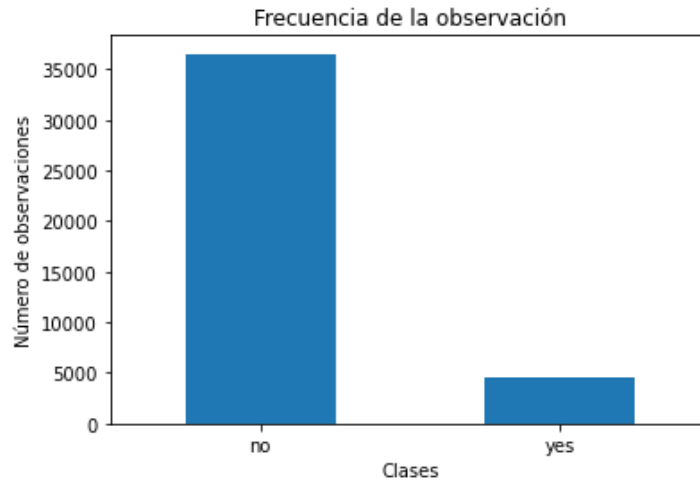
3. DEFAULT, LOAN Y HOUSING

Nos sirve para ver la distribución de personas.

4. VARIABLE 'Y'

Nos da cuenta de que esta dataset está completamente desbalanceado ya que al comparar 'yes' y 'no', nos encontramos con que tenemos más 'no' que 'sí'.





En el gráfico se evidencia que nuestros datos están desbalanceados, es decir, poseemos más de una clase que de otra: tenemos una clase minoritaria que en este caso es 'sí'. Esto provoca un desbalanceo en los datos que utilizaremos para el entrenamiento del modelo. Esta situación, por lo general, afecta a los algoritmos en su proceso de generalización de la información y perjudicando a las clases minoritarias. Esto lo debemos tratar de algún modo de cara al despliegue de nuestros modelos.



10. TRATAMIENTO DESBALANCEADOS

En los problemas de clasificación solemos encontrar que en nuestro conjunto de datos de entrenamiento contamos con que alguna de las clases de muestra es una clase “minoritaria” es decir, de la cual tenemos muy poquitas muestras. Esto provoca un desbalanceo en los datos que utilizaremos para el entrenamiento de nuestra máquina.

Por lo general afecta a los algoritmos en su proceso de generalización de la información y perjudicando a las clases minoritarias.

Si medimos la efectividad de nuestro modelo por la cantidad de aciertos que tuvo, sólo teniendo en cuenta a la clase mayoritaria podemos estar teniendo una falsa sensación de que el modelo funciona bien.

Estrategias para el manejo de Datos Desbalanceados:

Tenemos diversas estrategias para tratar de mejorar la situación. Las comentaremos brevemente y pasaremos a la acción (al código!) a continuación.

1. Ajuste de Parámetros del modelo:

Consiste en ajustar parametros ó metricas del propio algoritmo para intentar equilibrar a la clase minoritaria penalizando a la clase mayoritaria durante el entrenamiento. Ejemplos on ajuste de peso en árboles, también en logisticregression tenemos el parámetro `class_weight= “balanced”` que utilizaremos en este ejemplo. No todos los algoritmos tienen estas posibilidades. En redes neuronales por ejemplo podríamos ajustar la métrica de Loss para que penalice a las clases mayoritarias.

2. Modificar el Dataset:

Podemos eliminar muestras de la clase mayoritaria para reducirlo e intentar equilibrar la situación. Tiene como “peligroso” que podemos prescindir de muestras importantes, que brindan información y por lo tanto empeorar el modelo. Entonces para seleccionar qué muestras eliminar, deberíamos seguir algún criterio. También podríamos agregar nuevas filas con los mismos valores de las clases minoritarias, por ejemplo cuadruplicar nuestras 492 filas. Pero esto no sirve demasiado y podemos llevar al modelo a caer en overfitting.

3. Muestras artificiales:

Podemos intentar crear muestras sintéticas (no idénticas) utilizando diversos algoritmos que intentan seguir la tendencia del grupo minoritario. Según el método, podemos mejorar los resultados. Lo peligroso de crear muestras sintéticas es que podemos alterar la distribución



“natural” de esa clase y confundir al modelo en su clasificación.

4. Balanced Ensemble Methods:

Utiliza las ventajas de hacer ensamble de métodos, es decir, entrenar diversos modelos y entre todos obtener el resultado final (por ejemplo “votando”) pero se asegura de tomar muestras de entrenamiento equilibradas.



11. ALGORITMOS ELEGIDOS

En un comienzo, para comenzar a realizar nuestro proyecto desarrollamos 5 modelos de ML:

1. Decision Tree:

Los árboles de decisión son modelos predictivos formados por reglas binarias (si/no) con las que se consigue repartir las observaciones en función de sus atributos y predecir así el valor de la variable respuesta.

Aquí creamos efectivamente el árbol de clasificación. Vamos a empezar creando un árbol con una profundidad igual a 40 y luego vamos a jugar con los diferentes parámetros. Como todos los modelos en scikit-learn hay una línea que hace referencia a la construcción del modelo, otra al entrenamiento, y luego podemos predecir o ver el rendimiento del modelo.

2. Random Forest:

Es una combinación de árboles predictores tal que cada árbol depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para cada uno de estos.

3. K Nearest Neighbor (KNN):

Es un método que busca en las observaciones más cercanas a la que se está tratando de predecir y clasifica el punto de interés basado en la mayoría de datos que le rodean. Como dijimos antes, es un algoritmo:

- Supervisado: esto -brevemente- quiere decir que tenemos etiquetado nuestro conjunto de datos de entrenamiento, con la clase o resultado esperado dada “una fila” de datos.
- Basado en Instancia: Esto quiere decir que nuestro algoritmo no aprende explícitamente un modelo (como por ejemplo en Regresión Logística o árboles de decisión). En cambio memoriza las instancias de entrenamiento que son usadas como “base de conocimiento” para la fase de predicción.

1. Calcular la distancia entre el ítem a clasificar y el resto de ítems del dataset de entrenamiento.
2. Seleccionar los “k” elementos más cercanos (con menor distancia, según la función



que se use)

3. Realizar una “votación de mayoría” entre los k puntos: los de una clase/etiqueta que dominen decidirán su clasificación final.
4. Regresión Logística:

La regresión logística es un algoritmo de clasificación que se utiliza para asignar observaciones a un conjunto discreto de clases. Pertenece al aprendizaje automático que se utiliza para los problemas de clasificación, es un algoritmo de análisis predictivo y se basa en el concepto de probabilidad. La hipótesis de regresión logística tiende a limitar la función de costo entre 0 y 1.

5. Support Vector Machine:

El algoritmo de Support Vector Machine es encontrar un hiperplano en un espacio N-dimensional (N - el número de características) que clasifica claramente los puntos de datos.

Para separar las dos clases de puntos de datos, hay muchos hiperplanos posibles que podrían elegirse. Nuestro objetivo es encontrar un plano que tenga el margen máximo, es decir, la distancia máxima entre puntos de datos de ambas clases. Maximizar la distancia del margen proporciona cierto refuerzo para que los puntos de datos futuros se puedan clasificar con más confianza.

Posteriormente, sumamos métodos de ensamble. El ensamblado permite combinar un conjunto de diversos de modelos para improvisar sobre la estabilidad y el poder predictivo del modelo.

En sí, son algoritmos formados por algoritmos más simples. Estos algoritmos simples, se unen para formar un algoritmo más potente.

Aunque hay diversas formas de ensamblar o unir algoritmos débiles para formar otros, las más usadas y populares son el bagging y el boosting. Cada tipo de algoritmo tiene unas ventajas y desventajas, pudiendo ser usados convenientemente según sea nuestra problemática a resolver.

Haremos métodos de boosting porque bagging ya realizamos previamente: Random Forest.

Métodos de Boosting



En los algoritmos de boosting, los modelos simples son utilizados secuencialmente, es decir, cada modelo simple va delante o detrás de otro modelo simple. El principal objetivo de los métodos secuenciales es el de aprovecharse de la dependencia entre los modelos simples.

El rendimiento general puede ser mejorado haciendo que un modelo simple posterior, le dé más importancia a los errores cometidos por un modelo simple previo. Sería como si nosotros al resolver un problema, aprovechásemos nuestro conocimiento de los errores de otros para mejorar en algo intentando no cometerlos nosotros.

Las predicciones de cada modelo simple se combinan por medio de una votación en problemas de clasificación.

Algoritmos de ensamble:

1. XG BOOSTING

Extreme Gradient Boosting, es uno de los algoritmos de machine learning de tipo supervisado más usados en la actualidad. Este algoritmo se caracteriza por obtener buenos resultados de predicción con relativamente poco esfuerzo, en muchos casos equiparables o mejores que los devueltos por modelos más complejos computacionalmente, en particular para problemas con datos heterogéneos.

Se basa en el principio de “boosting”, que como bien sabemos, consiste en generar múltiples modelos de predicción “débiles” secuencialmente y que cada uno de estos tome los resultados del modelo anterior, para generar un modelo más “fuerte”, con mejor poder predictivo y mayor estabilidad en sus resultados.

Para conseguir un modelo más fuerte, se emplea un algoritmo de optimización, este caso Gradient Descent (descenso de gradiente). Durante el entrenamiento, los parámetros de cada modelo débil son ajustados iterativamente tratando de encontrar el mínimo de una función objetivo, que puede ser la proporción de error en la clasificación, el área bajo la curva (AUC), la raíz del error cuadrático medio (RMSE), etc.

Cada modelo es comparado con el anterior. Si un nuevo modelo tiene mejores resultados, entonces se toma este como base para realizar nuevas modificaciones. Si, por el contrario, tiene peores resultados, se regresa al mejor modelo anterior y se modifica ese de una manera diferente.



2. ADABOOST

Este algoritmo entrena de forma secuencial un conjunto de aprendices débiles a partir de un algoritmo base común. Todos los aprendices son entrenados con el mismo conjunto de datos pero éstos van recibiendo pesos que dependen de los errores cometidos por cada aprendiz. Así, inicialmente todas las muestras reciben un peso inicial w_i de $1/n$ (suponiendo que haya n muestras). El primer aprendiz es entrenado y se estima su tasa de error.

3. GRADIENT BOOSTING

El aumento de gradiente es un tipo de algoritmo de aumento. Se basa en la intuición de que el mejor modelo siguiente posible, cuando se combina con modelos anteriores, minimiza el error de predicción general.

La idea clave es establecer los resultados objetivo para este próximo modelo con el fin de minimizar el error. El aumento de gradiente se puede utilizar tanto para clasificación como para regresión.

4. LIGHT GBM

Utiliza la técnica Gradient Boosting. Con este método los árboles se construyen de manera secuencial y cada uno que se agrega aporta su granito de arena para refinar la predicción anterior. Es decir, se comienza con un valor constante y cada árbol nuevo se entrena para predecir el error en la suma de todas las predicciones de los árboles anteriores.

Una vez terminado el proceso, las predicciones se calculan sumando los resultados de todos los árboles que se construyeron. El efecto que tiene esto es que cada vez que se agrega un árbol nuevo se le presta atención a las muestras en las que el modelo está funcionando peor y se trabaja para mejorar ese aspecto.



12. MÉTRICAS DE DESEMPEÑO DEL MODELO

Al estar trabajando con un dataset desbalanceado, utilizamos principalmente el [f1_score](#) ya que en el análisis estadístico de la clasificación binaria, esta métrica es una medida de la precisión de una prueba. Se calcula a partir de la precisión y el recall de la prueba, donde la precisión es el número de resultados positivos verdaderos dividido por el número de todos los resultados positivos, incluidos los que no se identificaron correctamente, y el recuerdo es el número de resultados positivos verdaderos dividido por el número de todas las muestras que deberían haber sido identificadas como positivas. La precisión también se conoce como valor predictivo positivo y la recuperación también se conoce como sensibilidad en la clasificación binaria de diagnóstico.

Como dijimos, el valor F1 se utiliza para combinar las medidas de precisión y recall en un sólo valor. Esto es práctico porque hace más fácil el poder comparar el rendimiento combinado de la precisión y la exhaustividad entre varias soluciones.

F1 se calcula haciendo la media armónica entre la precisión y la exhaustividad:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

El valor F1 asume que nos importa de igual forma la precisión y la exhaustividad pero esto no tiene que ser así en todos los problemas igualmente.

Además, lo que hicimos fue realizar un [classification_report](#) de cada algoritmo corrido porque eso nos da un insight mayor del resultado que estamos obteniendo.



13. ITERACIONES DE OPTIMIZACIÓN.

Las principales iteraciones de optimización se realizaron a través de:

- Randomized Search: ajustamos hiper parámetros de un modo menos costoso que otros métodos ya que no es tan lento al momento de deployarlo. Buscamos los hiperparámetros más adecuados para nuestros modelos.
- Grid Search CV: ajustamos hiperparámetros de nuestros modelos para lograr una precisión aún mayor en el output. Primero hicimos un random search sobre los diferentes hiper parámetro, verificamos cuales son los hiper parámetros elegidos y luego sobre esos últimos pueden hicimos un grid search más fino alrededor de ellos.
- Realizamos K-Fold Cross Validation para evaluar los resultados de un análisis estadístico y garantizar que son independientes de la partición entre datos de entrenamiento y prueba ya que consiste en repetir y calcular la media aritmética obtenida de las medidas de evaluación sobre diferentes particiones. Se utiliza en entornos donde el objetivo principal es la predicción y se quiere estimar la precisión de un modelo que se llevará a cabo a la práctica.
- Tratamiento de desbalanceados: intentamos balancear el dataset ya que encontramos que la variable target tenía muchos más 'no' que 'sí', por lo que los modelos tendían a overfittear. Para esto realizamos:
 - *subsampling* (reducción clase mayoritaria);
 - *combinamos subsampling con Smote-Tomek* (aplica en simultáneo un algoritmo de subsampling y otro de oversampling a la vez al dataset. En este caso usaremos SMOTE para oversampling: busca puntos vecinos cercanos y agrega puntos "en linea recta" entre ellos. Y usaremos Tomek para undersampling que quita los de distinta clase que sean nearest neighbor y deja ver mejor el decisión boundary (la zona limítrofe de nuestras clases));
 - *oversampling* (consiste en modificar la distribución de los datos incrementando el número de casos de la clase minoritaria).



14. MÉTRICAS FINALES DEL MODELO OPTIMIZADO

Realizamos una curva ROC para analizar qué tan buenos clasificadores eran los modelos que realizamos.

ROC es un acrónimo que viene del inglés Receiver Operating Characteristic (Característica Operativa del Receptor). Es una gráfica que enfrenta el ratio de falsos positivos (eje x) con el ratio de falsos negativos (eje y). Estos ratios los va obteniendo en función de una serie de umbrales definidos entre 0 y 1. En palabras comunes y referenciando al ejemplo anterior, enfrenta la «falsa alarma» vs la tasa de éxito.

La tasa de falsos positivos se calcula como el número de positivos verdaderos divididos entre el número de positivos verdaderos y de falsos negativos. Describe cómo de bueno es nuestro modelo prediciendo las clases positivas cuando la salida real es positiva. También se conoce esta tasa como sensibilidad.

La tasa de falsos positivos se calcula como el número de falsos positivos dividido entre la suma de falsos positivos con los verdaderos negativos. Se considera como la tasa de «falsa alarma» y resumen como de común es que una clase negativa sea determinada por el modelo como positiva.

La especificidad es la inversa de la tasa de falsos positivos. Se obtiene dividiendo el número total de verdaderos negativos entre la suma de los verdaderos negativos y los falsos positivos.

La curva ROC es útil por dos principales motivos:

- Permite comparar diferentes modelos para identificar cual otorga mejor rendimiento como clasificador.
- El área debajo de la curva (AUC) puede ser utilizado como resumen de la calidad del modelo.

En resumen:

- Valores pequeños en el eje X indican pocos falsos positivos y muchos verdaderos negativos
- Valores grandes en el eje Y indican elevados verdaderos positivos y pocos falsos negativos

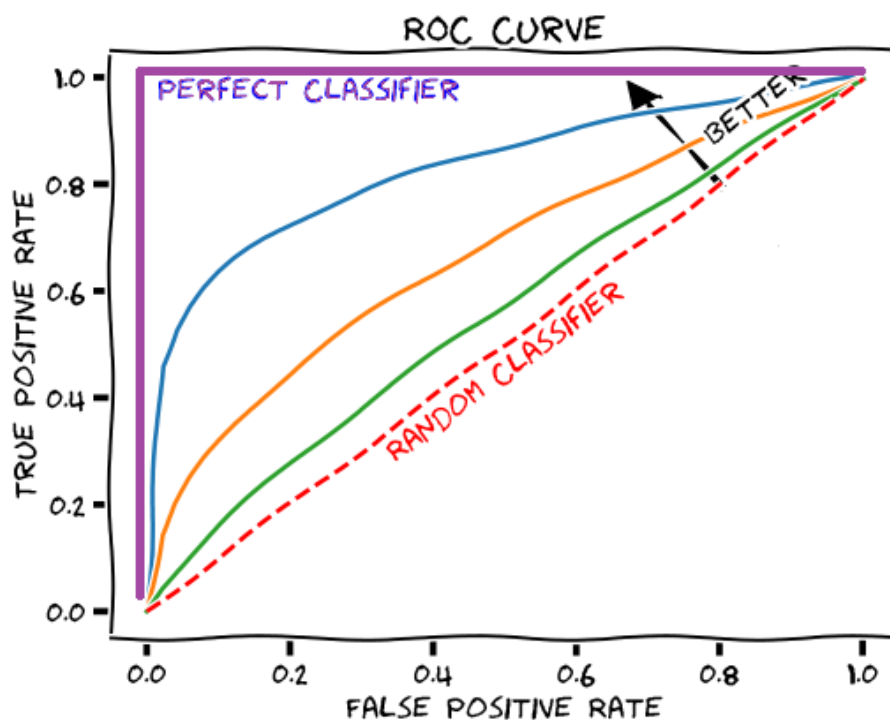


En el dibujo a continuación podemos observar la curva ROC de diversos modelos (cada color es un modelo). Un clasificador aleatorio (que asignará 0 o 1 al azar) estaría representado en rojo. Es decir, por azar, el 50% de las predicciones serían acertadas.

Según se desplaza la curva hacia la esquina superior izquierda del gráfico, la calidad del modelo va aumentando. Esto se debe a que mejora en su tasa de verdaderos positivos, minimizando también la tasa de falsos positivos.

El valor AUC se utiliza como resumen del rendimiento del modelo. Cuanto más esté hacia la izquierda la curva, más área habrá contenida bajo ella y por ende, mejor será el clasificador. El clasificador aleatorio tendría una AUC de 0.5 mientras que el clasificador perfecto (en morado) tendría un AUC de 1.

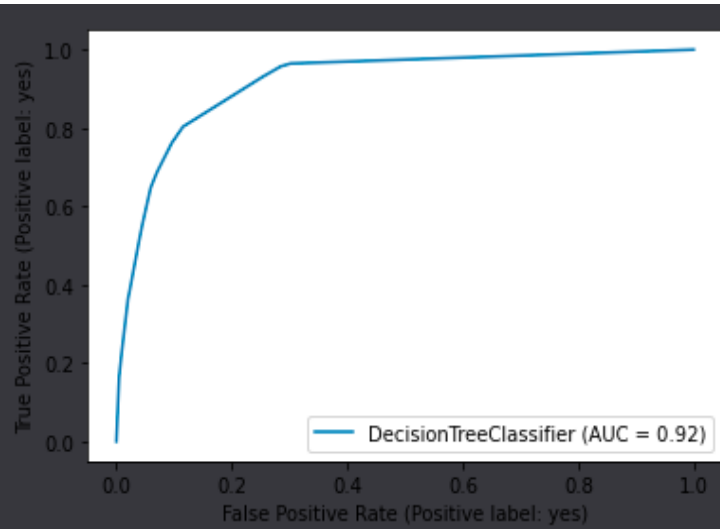
A la hora de desarrollar nuestro modelo, la curva ROC nos será de una gran utilidad. Nos permitirá elegir un umbral que optimice el comportamiento de nuestro modelo para resolver de la mejor manera nuestro problema de clasificación.



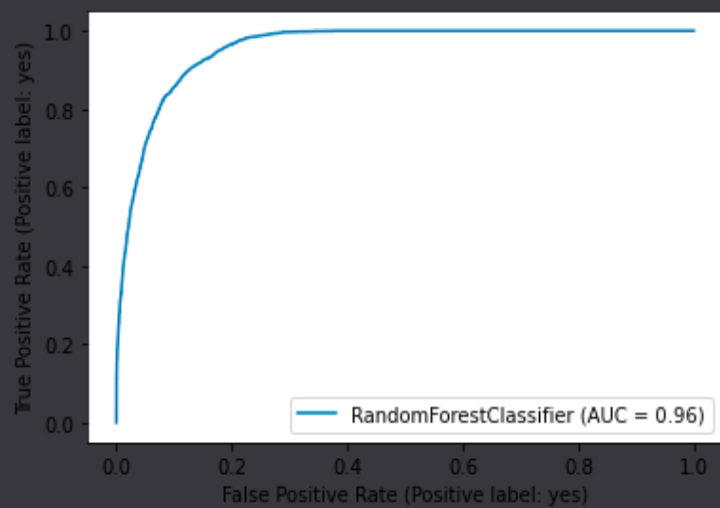
En nuestro proyecto, realizamos la curva ROC para cada uno de nuestros modelos de Machine Learning y estos fueron los resultados:



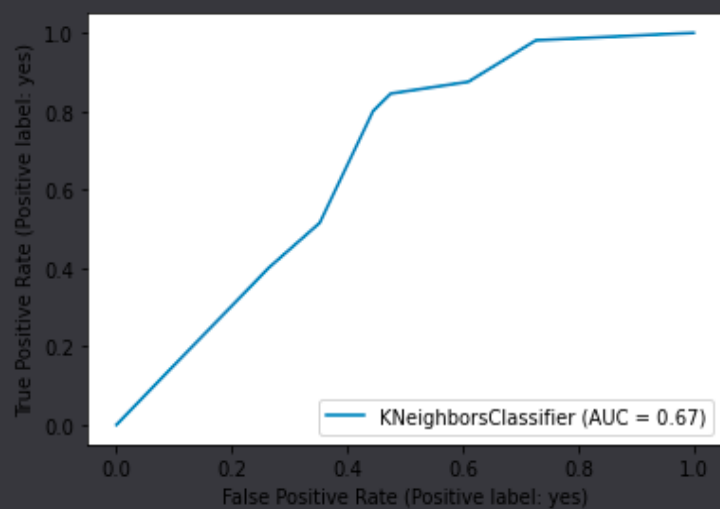
1. ÁRBOL DE DECISIÓN



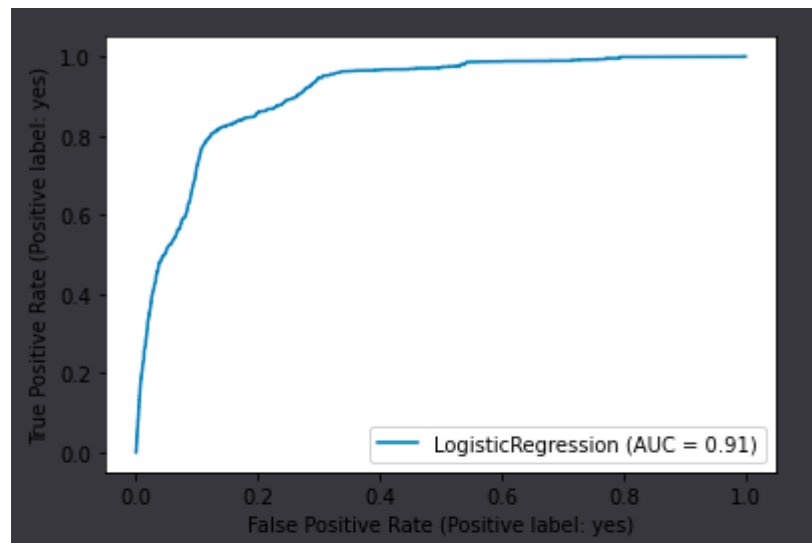
2. RANDOM FOREST



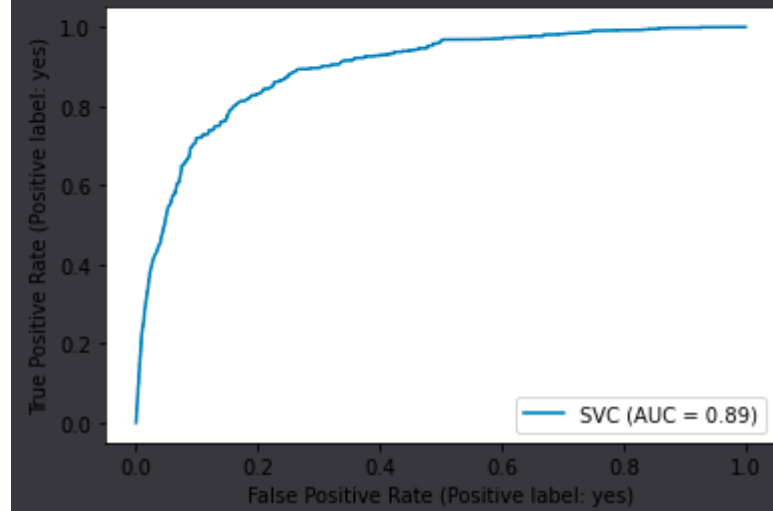
3. KNN



4. REGRESIÓN LOGÍSTICA



5. SVM



15. FUTURAS LÍNEAS: BREVE DESCRIPCIÓN DE LAS POSIBLES INICIATIVAS QUE SE PUEDEN LLEVAR A CABO PARA COMPLEMENTAR EL PROYECTO

Posteriormente se podría seguir trabajando en el desbalanceo del dataset, afinando algunas variables y seguir probando diferentes variables en los modelos, principalmente de ensamble o boosting. La idea a futuro sería implementar pipelines. Estos se utilizan para ayudar a automatizar los flujos de trabajo de aprendizaje automático. Operan al permitir que una secuencia de datos se transforme y correlacione en un modelo que se puede probar y evaluar para lograr un resultado, ya sea positivo o negativo.

Los pipelines de aprendizaje automático (ML) constan de varios pasos para entrenar un modelo. Los pipelines son iterativos ya que cada paso se repite para mejorar continuamente la precisión del modelo y lograr un algoritmo exitoso. Para crear mejores modelos de aprendizaje automático y sacar el máximo provecho de ellos, son imperativas soluciones de almacenamiento accesibles, escalables y duraderas, que allanan el camino para el almacenamiento de objetos en las instalaciones.

El objetivo principal de tener una canalización adecuada para cualquier modelo de ML es ejercer control sobre él. Una canalización bien organizada hace que la implementación sea más flexible.

El algoritmo de aprendizaje encuentra patrones en los datos de entrenamiento que asignan los atributos de los datos de entrada al objetivo (la respuesta a predecir) y genera un modelo ML que captura estos patrones. Un modelo puede tener muchas dependencias y para almacenar todos los componentes para asegurarse de que todas las funciones estén disponibles tanto fuera de línea como en línea para la implementación, toda la información se almacena en un repositorio central. En este caso, los pipelines constan de una secuencia de componentes que son una compilación de cálculos. Los datos se envían a través de estos componentes y se manipulan con la ayuda de la computación.

Los pipelines no son flujos unidireccionales, son de naturaleza cíclica y permiten la iteración para mejorar las puntuaciones de los algoritmos de aprendizaje automático y hacer que el modelo sea escalable.



16. CONCLUSIONES: CONTRIBUCIONES DEL MODELO A LOS OBJETIVOS PROPUESTOS

Para poder comenzar a armar las conclusiones del trabajo, primero es importante marcar la ruta de trabajo que se hizo sobre el proyecto:

EDA	Preparación Dataset	Machine Learning	Optimizaciones	Cierre
<ul style="list-style-type: none">• Análisis exploratorio y limpieza de datos• Análisis univariado• Análisis bivariado• Análisis multivariado• Desarrollo de visualizaciones y gráficos	<ul style="list-style-type: none">• Desglose del dataset• Preparación para utilizarlo como input en los diferentes modelos• Análisis de la variable target• Label Encoder• Tratamiento variables categóricas	<ul style="list-style-type: none">• Supervised Learning• Classification Models:<ul style="list-style-type: none">- Árbol de decisión- Random Forest- K-Nearest Neighbor- Regresión Logística- Support Vector Machine	<ul style="list-style-type: none">• Grid Search CV• Randomized Search• K-Fold Cross Validation• Ensemble Methods:<ul style="list-style-type: none">- XG Boosting- Ada Boost- Gradient Boosting- Light GBM- GaussianNB• Tratamiento de dataset desbalanceado:<ul style="list-style-type: none">- Subsampling en la clase mayoritaria por modelo- Combinamos resampling con Smote-Tomek- Oversampling de la clase minoritaria	<ul style="list-style-type: none">• Conclusiones finales del proyecto

El Exploratory Data Analysis nos permitió comprender la problemática que abordamos de mejor manera, entender las variables con las que tratamos y que importancia tenían para el modelo de negocio, siempre poniendo a este último como el principal foco de análisis a comprender para poder mejorar la performance de nuestros futuros modelos.

Posteriormente, preparamos el dataset y trabajamos sobre las variables categóricas. Pasamos todas estas a variables numéricas a través del Label Encoder y con ello utilizamos más features como input a nuestros modelos.

Luego realizamos 5 modelos los cuales nos dieron los siguientes valores en cuanto a su F1



Score, curva ROC y Cross Validation:

Modelo	Score Train Set	Score Test Set	Curva ROC	K-Fold Cross Validation
Árbol de Decisión	0.911	0.906	0.92	0.83 f1_score con una desviación estándar de 0.17
Random Forest	0.916	0.883	0.97	0.78 f1_score con una desviación estándar de 0.18
KNN	0.919	0.908	0.64	0.73 f1_score con una desviación estándar de 0.18
Regresión Logística	0.907	0.905	0.87	0.89 f1_score con una desviación estándar de 0.07
SVM		0.903	0.88	0.82 f1_score con una desviación estándar de 0.16

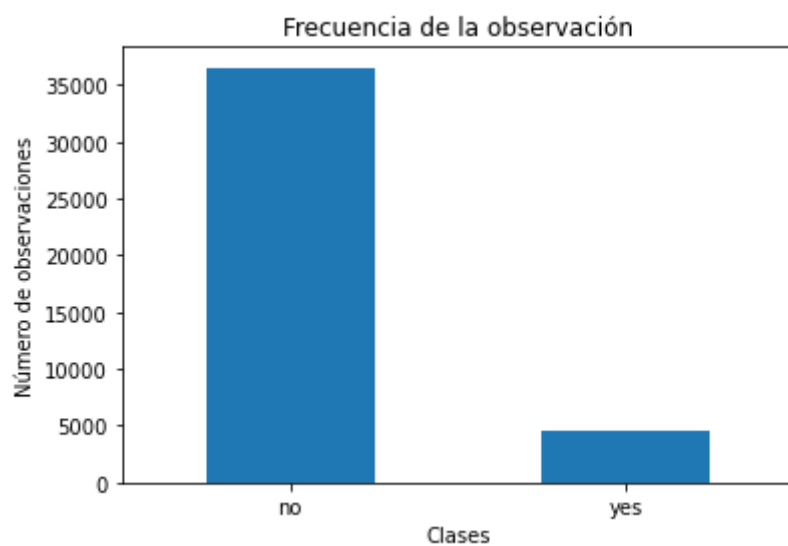
Y luego probamos con los modelos de ensamble que nos dieron los siguientes resultados:

Modelo	Score Train Set	Score Test Set	Curva ROC	K-Fold Cross Validation
XG Boost	0.920	0.911	0.94	0.75 f1_score con una desviación estándar de 0.22
AdaBoost		0.910	0.93	0.82 f1_score



				con una desviación estándar de 0.18
Gradient Boosting	0.917	0.912	0.94	0.76 f1_score con una desviación estándar de 0.20
Light GBM	0.926	0.912	0.94	0.70 f1_score con una desviación estándar de 0.22
GaussianNB		0.898	0.88	0.87 f1_score con una desviación estándar de 0.88

Por lo tanto, el modelo que mejor performó no fue únicamente por el simple hecho de tener una buena cantidad de aciertos en relación a predecir aquellos clientes que NO se suscriben a plazo fijo (mayoría) sino aquellos que SI lo hacen (minoría).



Por lo que, entre los que mejor performaron en base a lo anteriormente explicado están los siguientes modelos:



- Random Forest, método de bagging
- XG Boosting, método de ensamble
- AdaBoost, método de ensamble
- Gradient Boosting, método de ensamble
- Light GBM, método de ensamble.

Sus resultados prediciendo tanto los SI como NO fueron completamente superiores al del resto de los modelos. Estos son los resultados de los *classification_report* que hicimos y sus matrices de confusión:

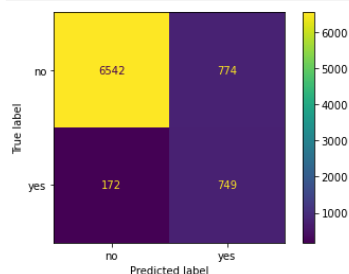
Random Forest:

Classification Report:

```
[38]: target_names = ['yes', 'no']
class_report = print(classification_report(y_pred_train, y_train, target_names=target_names))
```

	precision	recall	f1-score	support
yes	0.91	1.00	0.95	26711
no	0.96	0.58	0.72	6236
accuracy			0.92	32947
macro avg	0.94	0.79	0.84	32947
weighted avg	0.92	0.92	0.91	32947

```
[39]: plot_confusion_matrix(model_clf, X_test, y_test)
plt.show()
```



XG Boosting:

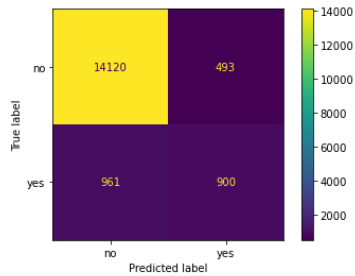


Classification Report:

```
[37]: target_names = ['yes', 'no']  
class_report = print(classification_report(y_train_pred, y_train, target_names=target_names))
```

	precision	recall	f1-score	support
yes	0.97	0.94	0.96	22666
no	0.52	0.70	0.59	2044
accuracy			0.92	24710
macro avg	0.74	0.82	0.78	24710
weighted avg	0.93	0.92	0.93	24710

```
[59]: # Matriz de confusión  
plot_confusion_matrix(clf_xgb, X_test, y_test)  
plt.show()
```



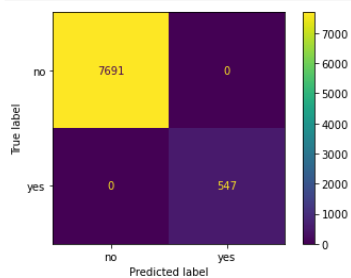
AdaBoost:

Classification Report:

```
[44]: target_names = ['yes', 'no']  
class_report = print(classification_report(y_train_pred, y_train, target_names=target_names))
```

	precision	recall	f1-score	support
yes	0.98	0.93	0.95	23132
no	0.38	0.68	0.49	1578
accuracy			0.91	24710
macro avg	0.68	0.80	0.72	24710
weighted avg	0.94	0.91	0.92	24710

```
[319]: # Siempre que hay un problema de clasificación, hacemos una matriz de confusión  
plot_confusion_matrix(ada, X_test, y_pred)  
plt.show()
```



Gradient Boosting:



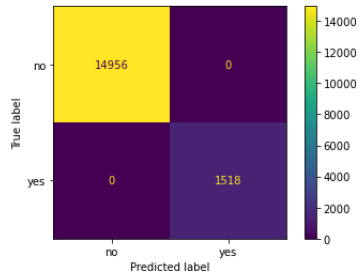
Classification Report:

```
[51]: target_names = ['yes', 'no']  
class_report = print(classification_report(y_train_pred, y_train, target_names=target_names))
```

	precision	recall	f1-score	support
yes	0.97	0.94	0.95	22435
no	0.54	0.66	0.60	2275
accuracy			0.92	24710
macro avg	0.75	0.80	0.78	24710
weighted avg	0.93	0.92	0.92	24710

```
[46]: # Siempre que hay un problema de clasificación, hacemos una matriz de confusión
```

```
plot_confusion_matrix(gbrt, X_test, y_pred)  
plt.show()
```



Light GBM:

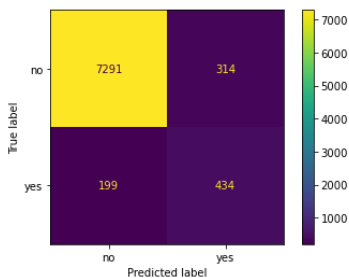
Classification Report:

```
[55]: target_names = ['yes', 'no']  
class_report = print(classification_report(y_train_pred, y_train, target_names=target_names))
```

	precision	recall	f1-score	support
yes	0.97	0.95	0.96	22567
no	0.56	0.73	0.63	2143
accuracy			0.93	24710
macro avg	0.77	0.84	0.80	24710
weighted avg	0.94	0.93	0.93	24710

```
[313]: # Siempre que hay un problema de clasificación, hacemos una matriz de confusión
```

```
plot_confusion_matrix(clf, X_test, y_pred)  
plt.show()
```



Como explicamos anteriormente, podríamos seguir el trabajo actual es seguir con la optimización de hiper parámetros, pero esta vez a través de pipelines.

