Módulo 3 - Aprendizaje supervisado: árboles de decisión

Introducción

En este módulo, conoceremos nuevos modelos de aprendizaje supervisado en *machine learning*, como son los árboles de decisión y sus diversos usos, que relacionan, incluso, nuestro día a día con el aprendizaje de máquinas.

Junto con ello, lograremos entender y diferenciar la manera en la que se realiza el aprendizaje de algoritmos para árboles de decisión, además de dividir, entrenar, visualizar e interpretar sus métricas. Para, posteriormente, aplicar procesamientos a los sets de datos, con el fin de que estos logren estar bien preparados y ajustados para su interpretación y uso.

En este espacio, se desarrollará la explicación detallada del contenido teórico-práctico. Es importante lograr que el estudiante encuentre relación entre los conceptos teóricos y su aplicación en su entorno profesional. Por esta razón, el uso de ejemplificaciones, infografía gráficos e imágenes es muy valioso.

Video de inmersión

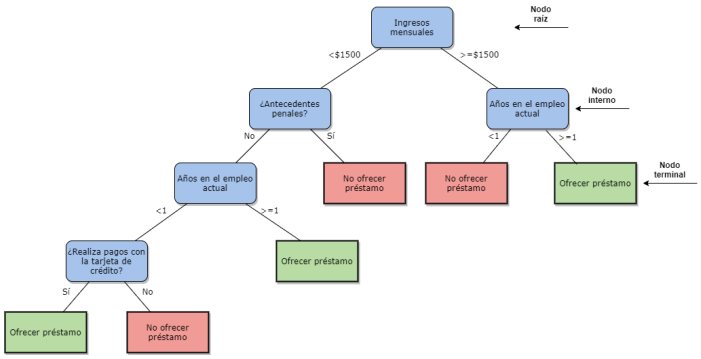
Unidad 1. Árboles de decisión

Tema 1. Árbol de decisión

Un árbol de decisión es un modelo de predicción utilizado en diversos ámbitos, que van desde la inteligencia artificial hasta la economía. Dado un conjunto de datos, se fabrican diagramas de construcciones lógicas, muy similares a los sistemas de predicción basados en reglas, que sirven para representar y categorizar una serie de condiciones que ocurren de forma sucesiva, para la resolución de un problema. (Wikipedia, s. f., https://es.wikipedia.org/wiki/%C3%81rbol_de_decisi%C3%B3n).

Los árboles de decisión son parte de los modelos predictivos clásicos de *machine learning*, la particularidad de estos es que están formados por reglas binarias (sí/no) [1 o 0] con las que se consigue repartir las observaciones en función de los atributos [de un set de datos] y así predecir el valor de una variable de [salida o] respuesta. (Amat, 2017, https://www.cienciadedatos.net/documentos/33_arboles_de_prediccion_bagging_random_forest_boosting).

Figura 1. Árbol de decisión



Fuente: Sanabria Castro, 2020, https://bit.ly/3Vzwz90

Los métodos basados en árboles se han convertido en uno de los referentes dentro del ámbito predictivo, debido a los buenos resultados que generan en problemas muy diversos.

Ventajas de un árbol de decisión:

- Los árboles son fáciles de interpretar, aun cuando las relaciones entre predictores son complejas.
- Los modelos basados en un solo árbol (no es el caso de *random forest, boosting*) se pueden representar gráficamente aun cuando el número de predictores es mayor a 3.
- Los árboles pueden, en teoría, manejar tanto predictores numéricos como categóricos, sin tener que crear variables *dummy o one-hot-encoding*. En la práctica, esto depende de la implementación del algoritmo que tenga cada librería.
- Al tratarse de métodos no paramétricos, no es necesario que se cumpla ningún tipo de distribución específica.
- Por lo general, requieren menos limpieza y preprocesado de los datos en comparación con otros métodos de aprendizaje estadístico (por ejemplo, no requieren estandarización).
- No se ven muy influenciados por *outliers*.
- Si, para alguna observación, el valor de un predictor no está disponible, a pesar de no poder llegar a ningún nodo terminal, se puede conseguir una predicción empleando todas las observaciones que pertenecen al último nodo alcanzado. La precisión de la predicción se verá reducida, pero al menos podrá obtenerse.
- Son muy útiles en la exploración de datos, permiten identificar de forma rápida y eficiente las variables (predictores) más importantes.
- Son capaces de seleccionar predictores de forma automática.
- Pueden aplicarse a problemas de regresión y clasificación. (Amat, 2017, https://www.cienciadedatos.net/documentos/33_arboles_de_prediccion_bagging_random_forest_boosting).

Desventajas de un árbol de decisión:

- La capacidad predictiva de los modelos basados en un único árbol es bastante inferior a la conseguida con otros modelos. Esto es debido a su tendencia al *overfitting* y alta varianza. Sin embargo, existen técnicas más complejas que, haciendo uso de la combinación de múltiples árboles (*bagging, random forest, boosting*), consiguen mejorar en gran medida este problema.
- Son sensibles a datos de entrenamiento desbalanceados (una de las clases domina sobre las demás).
- No son capaces de extrapolar fuera del rango de los predictores observado en los datos de entrenamiento. (Amat, 2020, https://www.cienciadedatos.net/documentos/33_arboles_de_prediccion_bagging_random_forest_boosting).

Árboles de decisión en Python

La principal implementación de árboles de decisión en Python está disponible en la librería Scikit learn a través de las clases *DecisionTreeClassifier y DecisionTreeRegressor*. Una característica importante para aquellos que han utilizado otras implementaciones es que, en Scikit learn, es necesario convertir las variables categóricas en variables *dummy (one-hot-encoding)*. (Amat, 2020, https://www.cienciadedatos.net/documentos/py07_arboles_decision_python.html).

Actividad de repaso

Una de las principales reglas para llevar a cabo un modelo de árbol de decisión pasa por convertir variables que son de tipo categóricas en

Archivos csv.

Variables no supervisadas.

Variables de analítica textual.

Variables dummy.

Variables cualitativas.

Justificación

Tema 2. Aplicaciones de un árbol de decisión

Dentro de este modelo de predicción, podemos encontrar diversas aplicaciones para la toma de decisiones, entre ellas, podemos mencionar:

1. Evaluación de oportunidades de crecimiento prospectivo

Una de las aplicaciones de los árboles de decisión consiste en evaluar las oportunidades de crecimiento prospectivo para las empresas, en función de los datos históricos. Los datos históricos sobre las ventas se pueden utilizar en árboles de decisión, que pueden llevar a realizar cambios radicales en la estrategia de una empresa para ayudar a la expansión y el crecimiento.

2. Uso de datos demográficos que permitan encontrar clientes potenciales

Otra aplicación de los árboles de decisión es el uso de datos demográficos para encontrar clientes potenciales. Pueden ayudar a optimizar un presupuesto de *marketing* y tomar decisiones informadas sobre el mercado objetivo en el que se centra la empresa. En ausencia de árboles de decisión, la empresa puede gastar su presupuesto de *marketing* sin tener en mente un grupo demográfico específico, lo que afectará sus ingresos generales.

3. Herramientas de apoyo para préstamos

Los bancos también usan árboles de decisión para predecir la probabilidad de que un cliente no cumpla con un préstamo, al aplicar la generación de modelos predictivos, utilizando los datos anteriores del cliente. El uso de una herramienta de soporte de árbol de decisiones puede ayudar a los prestamistas a evaluar la solvencia de un cliente para evitar pérdidas.

4. Investigación de operaciones

Los árboles de decisión también se pueden utilizar en la investigación de operaciones, en la planificación de la logística y la gestión estratégica.

Pueden servir para determinar las estrategias adecuadas que ayudarán a una empresa a alcanzar sus objetivos previstos.

5. Diagnóstico de enfermedades y dolencias

Los árboles de decisión pueden ayudar a los médicos y profesionales médicos a identificar a los pacientes que corren un mayor riesgo de desarrollar enfermedades graves (o prevenibles), como

diabetes o demencia. La capacidad de los árboles de decisión para reducir las posibilidades de acuerdo con variables específicas es muy útil en tales casos.

6. Detección de fraudes

Las empresas pueden prevenir el fraude utilizando árboles de decisión para identificar el comportamiento fraudulento de antemano. Puede ahorrar a las empresas una gran cantidad de recursos, incluidos tiempo y dinero.

Finalmente, dentro de todas las aplicaciones de los árboles de decisión, es importante poder dar respuesta a las siguientes preguntas:

¿Cuál es la característica más significativa en un algoritmo de árbol de decisión?

Los algoritmos de árboles de decisión son una herramienta valiosa para el análisis de decisiones y riesgos y, a menudo, se expresan como un gráfico o una lista de reglas. La sencillez de uso de los algoritmos de árboles de decisión es una de sus características más esenciales. Son fácilmente comprensibles y relevantes, ya que son visuales. Incluso, si los usuarios no están familiarizados con la construcción de algoritmos de árboles de decisión, pueden aplicarlos con éxito. Los algoritmos de árboles de decisión se emplean con mayor frecuencia para anticipar eventos futuros en función de la experiencia previa y ayudar en la toma racional de decisiones. Otro campo importante de los algoritmos de árboles de decisión es la minería de datos, donde los árboles de decisión se utilizan como una herramienta de clasificación y modelado, como se analiza más adelante.

¿Qué tan importante es un algoritmo de árbol de decisión?

Un algoritmo de árbol de decisiones tiene la importante ventaja de forzar el análisis de todos los resultados concebibles de una decisión y rastrear cada camino hacia una conclusión. Genera un estudio detallado de las implicaciones a lo largo de cada rama e indica los nodos de decisión que requieren más investigación. Además, a cada dificultad, ruta de decisión y resultado se le asigna un valor único mediante algoritmos de árboles de decisión. Este método destaca las rutas de decisión importantes, reduce la incertidumbre, elimina la ambigüedad y aclara las implicaciones financieras de los cursos de acción alternativos. Cuando la información objetiva no está disponible, los usuarios pueden usar algoritmos de árboles de decisión para poner las opciones

en perspectiva entre sí, para comparaciones simples, usando probabilidades para las circunstancias.

Actividad de repaso

Dentro del campo de la medicina, existe un algoritmo que puede ayudar a los médicos y profesionales médicos a identificar a los pacientes que corren un mayor riesgo de desarrollar enfermedades graves o prevenibles, como diabetes o demencia. ¿Mediante qué algoritmo de machine learning esto se puede llevar a cabo?

- Algoritmos dummy.
- Algoritmos de árbol de decisión.
- Algoritmos de carácter cualitativo.
- Algoritmos de analítica textual.
- Algoritmos no supervisados.

Justificación

Tema 3. División de un árbol de decisión

Hasta ahora, hemos aprendido la importancia que puede tener un árbol de decisión dentro de diversas industrias de trabajo.

A continuación, comenzaremos a preparar los datos para realizar nuestro primer modelo de árbol de decisión. Para ello, primeramente, haremos la división de datos en entrenamiento y testeo para modelos de aprendizaje supervisado, como es regresión logística y regresión lineal. Pero en el caso de un árbol de decisión, ¿de qué manera puedo entender la usabilidad que este puede tener en mi día a día?

A continuación, revisaremos algunos ejemplos que nos permitirán entender cómo influyen los árboles cuando queremos tomar decisiones dentro de nuestra vida cotidiana en temas como, por ejemplo:

¿Ver una película o hacer un trabajo del módulo 3 de *machine learning*?

Para ello, empezamos a pensar si lo que decidimos es estrictamente algo que tenemos que hacer

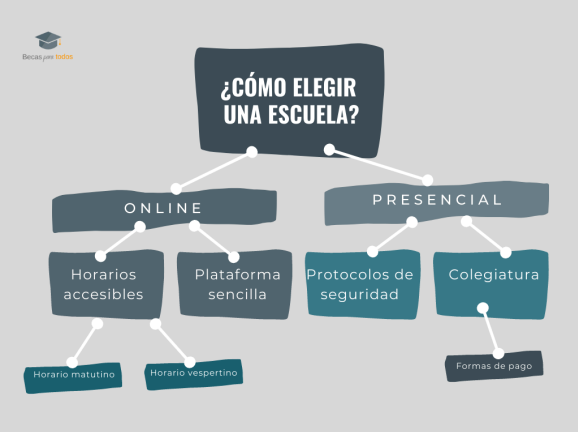
y qué se debe priorizar: ¿nuestra felicidad instantánea o nuestra responsabilidad?

En este sentido, entenderemos que nuestra mente actúa como un árbol de decisión, el cual es una forma gráfica que tiene una manera de dividir los sucesos con el fin de tomar una decisión.

¿Cómo podemos representarlo gráficamente?

Considera el ejemplo de ¿cómo elegir una escuela?

Figura 2. Ejemplo de árbol de decisión



Fuente: [Imagen sin título sobre ejemplo de árbol de decisión]. (s. f.). https://bit.ly/3Da8kXF

Este sería un caso que podemos resolver con un modelo de árbol de decisión y, posteriormente, evaluar cada una de las diferentes ramas, es decir, diferentes decisiones para que, al final, en los pequeños cuadros que se ven, se vea reflejada la posible decisión que tenemos que llevar a cabo.

Lo mismo ocurre con los casos como: si quieres ver televisión o si no quieres ver televisión, si se hace una tarea o no se hace esa tarea.

Cuando el árbol se empieza a dividir, comenzamos a tener respuestas tanto positivas como negativas.

Ahora, consideremos que cada uno de estos árboles creados puede tener ciertas ventajas y desventajas a nivel de *machine learning*.

Tabla 1. Ventajas y desventajas de árboles de decisión a nivel de *machine learning*

Ventajas	Desventajas
Claridad de los datos, podemos ver claramente los caminos que se pueden tomar.	Si no se definen bien ciertos criterios, la división del árbol puede ser deficiente.
Permite hacer predicciones bajo ciertas reglas extraídas.	Si se manejan pocos datos, se puede tener sobreajuste.
Tolerancia cuando existen datos faltantes, aunque esto no sea recomendable, podemos, de todas formas, realizar nuestro análisis.	Puede aparecer división del árbol en ramas poco significativas.

Fuente: elaboración propia.

Regularmente, un árbol de decisión se usa para saber cuál es la probabilidad de que una persona compre un artículo.

Por ejemplo, Amazon, a través de cierta información que ha ido recolectando en su base de datos de clientes, puede llegar a estimar nuestra decisión ante la posible compra de una cámara de fotos o no. Debido a que esto ya lo aprendió anteriormente de otras personas y dadas las características, puede discriminar, con base en nuestro nivel de compra, horarios de conexión a la plataforma, edad, género, etc.

Esta información se va dividiendo de acuerdo con estos criterios, que son los datos, quizás, más significativos, pero a medida que vamos teniendo más información, también se pueden crear árboles más pequeños para lograr una predicción.

Cabe recordar que, en todo tipo de modelo de *machine learning*, se debe evitar el sobreajuste de datos, lo cual puede perjudicar el aprendizaje del modelo, para ello, siempre es importante preprocesar la base de datos, de manera de tener:

- Variables de salida equilibradas (*output*).
- Imputar datos nulos, debido a que pueden generar ruido dentro del árbol, lo cual puede afectar a la división de la información.

Actividad de repaso

La empresa Sail X, dedicada a la importación de cosméticos, está intentando aplicar modelos de machine learning que le permitan decidir,

dentro sus estrategias, cuáles son las mejores opciones para importar sus productos. ¿Enviar sus productos a Brasil o España? Para ello, es recomendable hacer uso de un modelo:

- KNN.
- K-means.
- Árbol de decisión.
- Regresión multilineal.
- Regresión multilogística.

Justificación

Tema 4. Comprendiendo los *dataset* para la creación de un árbol de decisión

Para poder realizar nuestro primer árbol de decisión, haremos uso de datos de un evento histórico ocurrido hace más de 100 años: el viaje del Titanic. En este caso, usaremos una base de datos que consta de la lista de pasajeros que estuvieron en el viaje del 15 de abril de 1912, y que tienen distintas características.

Al final de nuestro ejercicio, podremos medir indicadores para ver con qué precisión somos capaces de acertar si el pasajero pudo sobrevivir o no pudo sobrevivir.

Para ello, utilizaremos una muestra de la base de datos con dos archivos CSV, *Titanic_Train, y Titanic_test.*

En el siguiente enlace, podemos encontrar el data set relacionado con la lista de pasajeros del Titanic:

https://www.kaggle.com/c/titanic/data

El conjunto de entrenamiento *Titanic_Train:* debe usarse para construir los modelos de aprendizaje automático. Para el conjunto de entrenamiento, proporcionamos el resultado (también conocido como "verdad") para cada pasajero. El modelo se basará en características como el género y la clase de los pasajeros, así como también otras características.

El conjunto de prueba *Titanic_test:* debe usarse para ver qué tan bien se desempeña el modelo

en datos no vistos. Para el conjunto de prueba, no proporcionamos los resultados finales para cada pasajero, ya que es nuestro trabajo predecir estos resultados. Para cada pasajero en el conjunto de prueba, se usará el aprendizaje del modelo que se entrenó para predecir si sobrevivieron o no al hundimiento del Titanic.

Para ello, comenzaremos trabajando con los siguientes pasos de cálculo:

Figura 3. Paso 1 - Importar librerías que se van a utilizar

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import re
import numpy as np
from sklearn import tree
from sklearn.model_selection import train_test_split

%matplotlib inline
sns.set()
```

Fuente: elaboración propia.

Importante: recuerda que, en todo modelo, primero, se debe entrenar y, luego, se debe poner a prueba mediante test, por lo que es fundamental contar con esta línea de código en cualquier tipo de modelo que uno quiera realizar.

Paso 2 - Cargar los datos

Lo siguiente que vamos a hacer es, justamente, cargar nuestra información, cargar nuestros datos que ya tenemos disponibles y visualizar las 5 primeras filas mediante la función head.

Figura 4. Carga de datos

```
test_df = pd.read_csv('/content/titanic-test.csv')
train_df = pd.read_csv('/content/titanic-train.csv')
train_df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Fuente: elaboración propia.

Figura 5. Paso 3 - Revisar cuánta información tiene almacenada el set de datos

```
print('La cantidad de datos para la variable test son',test_df.shape)
print('La cantidad de datos para la variable ejtrenamiento son',train_df.shape)

La cantidad de datos para la variable test son (418, 11)
La cantidad de datos para la variable ejtrenamiento son (891, 12)
```

Fuente: elaboración propia.

Revisaremos también si, para la base de entrenamiento, las variables contienen valores nulos y su tipo de dato.

Figura 6. Control de variables

```
1  train_df.info()
```

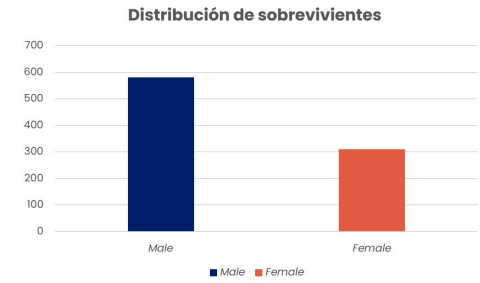
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

Fuente: elaboración propia.

También, revisaremos la distribución de sobrevivientes que contiene la base de datos, con la finalidad de ver qué tan balanceada puede estar nuestra base de datos en relación con sí sobrevivieron más hombres que mujeres o viceversa.

Figura 7. Revisión de distribución de sobrevivientes

```
1  train_df.Sex.value_counts().plot(kind = 'bar', color = ['b', 'r'])
2  plt.title('Distribucion de sobrevivientes')
3  plt.show()
```



Fuente: elaboración propia.

¿Qué es un conjunto de datos balanceados?

Dentro de todos los algoritmos de *machine learning*, siempre encontraremos información con datos que son de carácter desbalanceado, pero ¿qué significa que los datos sean desbalanceados?

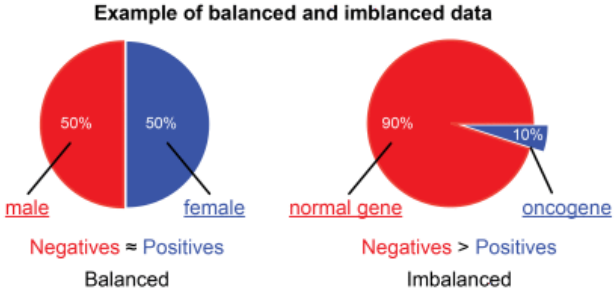
Esto quiere decir que nuestras etiquetas de respuesta o de salida de una base de datos se encuentran desequilibradas.

Aquí, un ejemplo de datos que pueden ser balanceados y desbalanceados.

Conjunto de datos balanceados: ocurre cuando, en un conjunto de datos, tenemos valores positivos que son aproximadamente iguales a los valores negativos. Entonces, podemos decir que nuestro conjunto de datos está balanceado.

Conjunto de datos desbalanceados: si existe una diferencia muy alta entre los valores positivos y los valores negativos. Entonces, podemos decir que nuestro conjunto de datos está desbalanceado.

Figura 8. Ejemplos



Fuente: [Imagen sin título sobre datos balanceados y desbalanceados]. (s. f.). https://bit.ly/3s6EwoE

¿Qué desventaja tiene trabajar con una base de datos desbalanceada?

Trabajar con datos desbalanceados, por ejemplo: tener 90 % de los datos con una variable de salida Yes y 10 % con variable de salida No puede generar que la máquina aprenda de información sobreajustada, lo que quiere decir, en otras palabras, que no tiene la precisión correcta para poder decidir entre una opción *Yes* y *No*. Entonces, a la hora de decidir por una opción, va a elegir *Yes*, debido a que la base de datos tiene una tendencia a esta respuesta. Esto podemos corroborarlo, por ejemplo, cuando los valores del indicador *accuracy* son del 100 %.

Es por eso que se recomienda que el balanceo de una base de datos sea lo más cercano a 50 % para cada una de las variables dicotómicas de salida.

Actividad de repaso

La empresa de embarques Embarcard quiere realizar análisis predictivos para la toma de decisiones estratégicas. Ante ello, han decido utilizar árboles de decisión que permitan precisar si es conveniente invertir más en nuevos buques de carga. Al revisar los datos, se han encontrado con

que el 80 % de estos corresponde a buques que no lograron ser cargados en su totalidad. El 20 % restante son buques que sí se lograron cargar. Según las características nombradas de la BBDD, podemos considerar que:

La base de datos se encuentra desbalanceada.

La base de datos debe usar modelos de aprendizaje no supervisado.

El posicionamiento de la marca está en riesgo, según los buques no cargados.

Se debe hacer un estudio de mercado para entender las solicitudes de pedidos de los clientes.

Justificación

Unidad 2. Clasificador de árboles y métricas

Tema 1. Clasificador de árboles con Scikit learn

En la sección anterior, trabajamos un set de datos en relación con la lista de pasajeros que fueron parte del viaje del Titanic de 1912. Con base en esta información, continuaremos explorando el set de datos para poder realizar nuestra primera clasificación de árboles de decisión.

En esta ocasión, podremos contar con preprocesamiento para trabajar los datos y llegar a resultados que sean un poco más complejos de los que anteriormente hemos revisado.

Esto se debe a que la base de datos contiene mucha información que necesitaremos dividir para poder aprender de aquellos datos que sean realmente útiles.

Recordemos que la mejor manera de generar un modelo de *machine learning* es con aquellas variables que realmente aportan valor al algoritmo de estudio, con lo cual no toda la información que encontremos dentro de una base será siempre la que tendremos que utilizar.

Para realizar esto, importaremos una nueva sección de librerías de Scikit learn.

Figura 9. Nueva sección de librerías de Scikit learn

```
1 from sklearn import preprocessing
2 label_encoder = preprocessing.LabelEncoder()
3
4 encoder_sex = label_encoder.fit_transform(train_df['Sex'])
5 train_df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Fuente: elaboración propia.

En la línea 1, encontraremos que se importa *preprocessing*, con la cual podremos preprocesar nuestro *data set* más adelante.

En la línea 2, se llama a *label_encoder*, la cual codifica etiquetas de una característica categórica en valores numéricos entre 0 y 1. Por el momento, la estamos definiendo, por lo que las transformaciones ocurrirán después.

En la línea 4, *encoder_sex* nos permitirá modificar la variable sex para poder trabajar con esta información de mejor manera (de una manera categórica).

Como siguiente paso del proceso, revisaremos si nuestro *data set* contiene datos nulos que tengamos que reemplazar.

Figura 10. Revisión de *data set*

```
1 train_df.isna().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

Fuente: elaboración propia.

Al revisar el resultado en la línea de código, nos damos cuenta de que la variable Age tiene 177 registros vacíos; la variable *Cabin*, 687; y la variable *Embarked*, 2 datos nulos.

Ahora, comenzaremos a rellenar los valores que no tienen datos para la variable *Age* (edad).

En este caso, para no perder información, a los valores nulos los rellenaremos con la mediana

estadística de edad que contiene la variable *Age*.

En el caso de *Embarked* (embarque), también realizaremos el mismo proceso.

Figura 11. Variables *Age* y *Embarked*

```
train_df['Age'] = train_df['Age'].fillna(train_df['Age'].median())
train_df['Embarked'] = train_df['Embarked'].fillna('S')
```

Fuente: elaboración propia.

Ahora trabajaremos con nuestros predictores, en los cuales tendremos que hacer unos cambios en los datos y, dentro de ellos, tendremos que eliminar aquellos datos que no nos agregan valor, mediante la función *Drop*.

Para ello, eliminaremos las variables *PassengerId, Survived, Name, Ticket y Cabin.*

Ojo al dato:

Survived se elimina porque es la etiqueta que queremos que el algoritmo busque predecir.

Posteriormente, tomaremos los siguientes datos categóricos y vamos a crear una variable adicional de datos solo categóricos.

Figura 12. Creación de variable adicional de datos categóricos

```
train_predictors = train_df.drop(['PassengerId', 'Survived', 'Name', 'Ticket', 'Cabin'], axis = 1)

categorical_cols = [cname for cname in train_predictors.columns if
                    train_predictors[cname].nunique() < 10 and
                    train_predictors[cname].dtype == 'object'
                    ]

numerical_cols = [cname for cname in train_predictors.columns if
                  train_predictors[cname].dtype in ['int64', 'float64']
                  ]
```

Fuente: elaboración propia.

Figura 13. Revisamos cada una de las variables

```
1  train_predictors
```

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	male	22.0	1	0	7.2500	S
1	1	female	38.0	1	0	71.2833	C
2	3	female	26.0	0	0	7.9250	S
3	1	female	35.0	1	0	53.1000	S
4	3	male	35.0	0	0	8.0500	S
...	...	...	...	...	...	...	...
886	2	male	27.0	0	0	13.0000	S
887	1	female	19.0	0	0	30.0000	S
888	3	female	28.0	1	2	23.4500	S
889	1	male	26.0	0	0	30.0000	C
890	3	male	32.0	0	0	7.7500	Q

891 rows × 7 columns

Fuente: elaboración propia.

Luego, realizaremos el mismo proceso, pero ahora para las columnas de carácter numérico, que, en este caso, son: *PClass, Age, SibSp, Parch, Fare*.

Figura 14. Creación de variables para las columnas de carácter numérico

```
numerical_cols = [cname for cname in train_predictors.columns if
                  train_predictors[cname].dtype in ['int64', 'float64']
]
```

```
numerical_cols
```

```
['Pclass', 'Age', 'SibSp', 'Parch', 'Fare']
```

Fuente: elaboración propia.

Una vez realizados estos procesos, haremos la consolidación de las variables categóricas y numéricas, mediante el siguiente proceso.

Figura 15. Consolidación de las variables categóricas y numéricas

```
my_cols = categorical_cols + numerical_cols

train_predictors = train_predictors[my_cols]
```

```
train_predictors
```

	Sex	Embarked	Pclass	Age	SibSp	Parch	Fare
0	male	S	3	22.0	1	0	7.2500
1	female	C	1	38.0	1	0	71.2833
2	female	S	3	26.0	0	0	7.9250
3	female	S	1	35.0	1	0	53.1000
4	male	S	3	35.0	0	0	8.0500
...	...	...	...	...	...	...	...
886	male	S	2	27.0	0	0	13.0000
887	female	S	1	19.0	0	0	30.0000
888	female	S	3	28.0	1	2	23.4500
889	male	C	1	26.0	0	0	30.0000
890	male	Q	3	32.0	0	0	7.7500

891 rows × 7 columns

Fuente: elaboración propia.

Teniendo el set de datos ya consolidado, haremos uso de variables artificiales conocidas como *Dummies*, que tienen como finalidad generar nuevas columnas de datos por cada una de las categorías que tenga cada variable original.

Por ejemplo, si la variable Sexo contiene *masculino y femenino*, la variable *dummy* generará dos columnas en el set de datos: una denominada Sexo Masculino y otra denominada Sexo Femenino, de la siguiente forma.

Figura 16. Variable *dummy*

```
dummy_encoded_train_predictors = pd.get_dummies(train_predictors)
```

```
dummy_encoded_train_predictors
```

	Pclass	Age	SibSp	Parch	Fare	Sex_female	Sex_male	Embarked_C	Embarked_Q	Embarked_S
0	3	22.0	1	0	7.2500	0	1	0	0	1
1	1	38.0	1	0	71.2833	1	0	1	0	0
2	3	26.0	0	0	7.9250	1	0	0	0	1
3	1	35.0	1	0	53.1000	1	0	0	0	1
4	3	35.0	0	0	8.0500	0	1	0	0	1
...	...	...	...	...	...	...	...	...	...	...
886	2	27.0	0	0	13.0000	0	1	0	0	1
887	1	19.0	0	0	30.0000	1	0	0	0	1
888	3	28.0	1	2	23.4500	1	0	0	0	1
889	1	26.0	0	0	30.0000	0	1	1	0	0
890	3	32.0	0	0	7.7500	0	1	0	1	0

891 rows × 10 columns

Fuente: elaboración propia.

Hasta esta parte, ya contamos con un set de datos preprocesado para ser trabajado posteriormente y generar nuestro árbol de decisión.

Actividad de repaso

¿A cuál de los siguientes conceptos corresponde la siguiente aseveración? Es una función que permite eliminar la primera de las columnas generadas para cada característica codificada para evitar la denominada colinealidad (que una de las características sea una combinación lineal de las otras), lo que dificulta el correcto funcionamiento de los algoritmos.

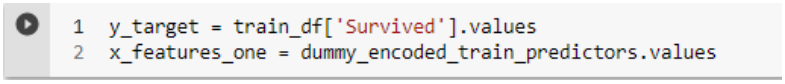
- Variables Pandas.
- Variables Random.
- Variables Anacondas.
- Variables regresoras.
- Variables dummies.

Justificación

Tema 2. Entrenamientos del modelo de clasificación

Para comenzar con el proceso de entrenamiento, primeramente, dividiremos nuestro set de datos en las columnas con las cuales aprenderá el modelo, denominaremos esto x_features_one, mientras que, por otro lado, tomaremos la variable objetivo que tiene los datos de salida y_target. No olvidar que esta variable es la que nos indica si la persona sobrevivió o no.

Figura 17. Variable y *_target*



Fuente: elaboración propia.

Ahora viene una parte totalmente importante, que es la de entrenar la información.

Para ello, se crearán 4 variables:

- *X_train:* entrenamiento para variable X.
- *X_validation:* test para variable X.
- *Y_train:* entrenamiento para variable Y.
- *Y_validation:* test para variable Y.

A cada uno de estos datos los vamos a dividir en entrenamiento y validación con el método: *train_test_split*

Según muestra el siguiente código.

Figura 18. Método *train_test_split*

Fuente: elaboración propia.

Ahora creamos nuestro primer árbol.

Figura 19. Creación de árbol

Fuente: elaboración propia.

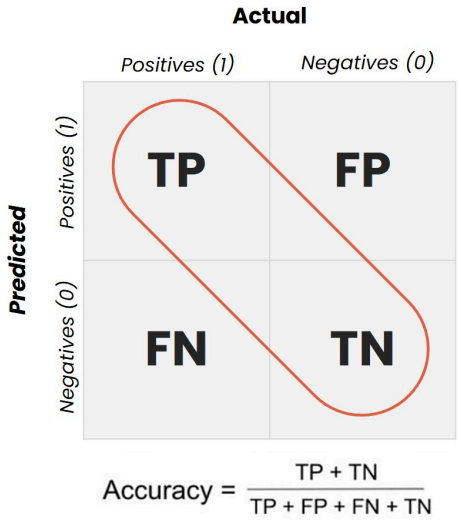
Posteriormente, tenemos que validar qué tan bien está entrenado nuestro árbol, haciendo uso de la precisión de información. Para ello, usaremos el indicador *accuracy.*

Recuerda

Accuracy: con la métrica de precisión podemos medir la **calidad** del modelo de *machine learning* en tareas de clasificación, este indicador podrá entregarnos como resultado el porcentaje total de elementos clasificados correctamente.

Aplicado, tiene un nivel de probabilidad alto de acertar a las predicciones de datos nuevos que se agreguen a la base de datos.

Figura 20. *Accuracy*



Fuente: elaboración propia.

Si aplicamos esta información a nuestro modelo, podemos ver que nuestro resultado es de un 74 %.

Figura 21. Aplicación de *accuracy*

Fuente: elaboración propia.

Esto quiere decir que es un muy buen modelo e indica que, cuando llegue un nuevo valor, este tiene un 74 % de probabilidades de que la etiqueta de salida que se le asigne sea correcta.

Actividad de repaso

¿Mediante qué indicador podemos medir la probabilidad de que la etiqueta de salida de un set de datos entrenados tenga una salida correcta?

- Awareness.
- Accuracy.
- F1-Score.
- Train.
- Test.

Justificación

Tema 3. Visualización de un árbol de decisión

En la sección anterior, vimos cómo crear el árbol de decisión basado en el entrenamiento del set de datos del Titanic.

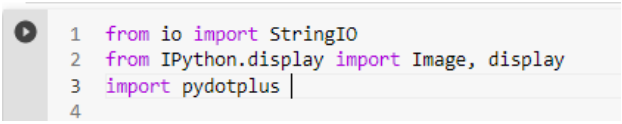
La pregunta que surge ahora es: ¿de qué nos sirve entrenar el modelo si no lo podemos visualizar? Entonces, en este momento es cuando realizaremos nuestra visualización de un árbol de decisión con base en el entrenamiento que hemos entregado a una máquina.

Para ello, recordemos que, al entrenar el modelo de Titanic, obtuvimos como resultado de precisión un 74 %.

Ahora, podremos observar cada uno de los caminos mediante los cuales la máquina aprende para poder tomar la decisión.

Para ello, utilizaremos el siguiente código, que nos permitirá mostrar cada uno de los caminos con los cuales el árbol es capaz de ir tomando decisiones a medida que va analizando cada uno de los datos.

Figura 22. Importación de la función StringIO



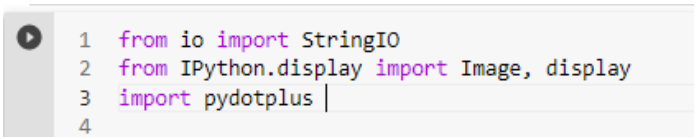
Fuente: elaboración propia.

En la línea 1, importaremos la función *StringIO*, la cual nos permite trabajar con diversos archivos externos (recuerda que estamos utilizando archivos .csv externos al entorno de Python).

En la línea 2, importamos de *Image y display*, esto nos da la posibilidad de interactuar y crear imágenes de tipo nodo.

Adicional a eso, también importaremos *pydotplus*, la cual nos permitirá usar el lenguaje *grapghviz* para crear imágenes.

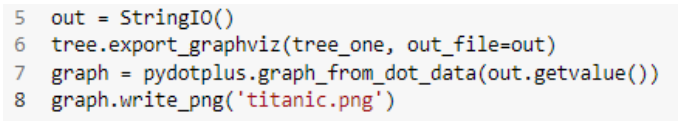
Figura 23. Importación de pydotplus



Fuente: elaboración propia.

Posteriormente, comenzaremos a ejecutar códigos que nos permitan generar el gráfico de nuestro primer árbol de decisión.

Figura 24. Ejecución de código para generar árbol de decisión



Fuente: elaboración propia.

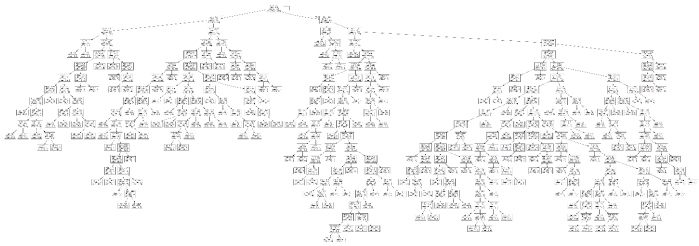
En la línea 5, exportaremos los datos del árbol en lenguaje *graphviz* a *StringIO*.

En la línea 7, generaremos el gráfico a través de *pydotplus*.

En la línea 8, guardamos el archivo en formato *png* para poder descargar.

Una vez ejecutado el código, podrás encontrar la impresión del árbol para visualizar.

Figura 25. Árbol generado

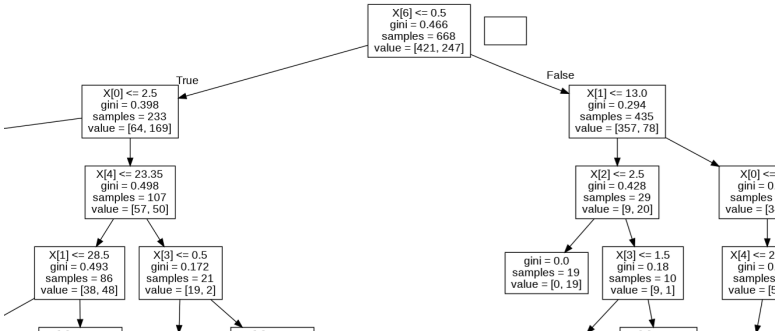


Fuente: elaboración propia.

Como podemos ver, el árbol contiene muchísimas ramas que representan cada una las decisiones y caminos que explican el aprendizaje de la máquina y por qué cada uno de los datos forma parte de cada una de esas ramas.

Para ello, haremos *zoom* en los nodos principales, para poder entender cada uno de estos datos.

Figura 26. Detalle del árbol generado



Fuente: elaboración propia.

En él podrás ver, por ejemplo:

- *Sample:* representa el 75 % de la muestra que tenía el set de datos original. Por ello es que el valor es de 668.
- *Value:* si sumamos 421 + 247, de los 668 datos de la muestra. Aquellos que tienen valor 1 (true) son 421, mientras que los valores 0 (falsos) son 247.

Actividad de repaso

Modelo de predicción que consta de ramas de información que representan cada una las decisiones y caminos que explican el aprendizaje de una máquina, ¿de qué modelo estamos hablando?

Regresión multilineal.

Regresión multilogística.

Árbol de decisión.

K-means.

KNN.

Justificación

Tema 4. Métricas y validación

Cuando necesitamos evaluar el rendimiento en clasificación, podemos usar las métricas de *accuracy, recall*, F1 y la matriz de confusión. Para ello, explicaremos la importancia de cada una de ellas y su utilidad práctica con un ejemplo.

Imaginemos el siguiente ejemplo:

Dentro de tu empresa, estás llevando la campaña de marketing para un banco. Al banco le interesa vender un fondo de inversión a sus clientes, porque así pueden ganar dinero por la comisión de gestión.

Para lograr esto, podríamos contactar con todos los clientes del banco y ofrecerles el fondo de inversión. Esto, quizás, podría generar un esfuerzo operativo que es bastante ineficiente, porque la mayoría de los clientes no estarán interesados. Sería más eficiente contactar con unos pocos, recoger datos, hacer *machine learning* y predecir qué otros clientes tienen más probabilidad de aceptar la oferta del banco.

Sin embargo, ningún modelo de *machine learning* es perfecto. Esto quiere decir que:

- Habrá clientes con los que contactaremos porque el modelo ha predicho que aceptarían y, en realidad, no lo harán (False Positive FP, positivos falsos).
- Habrá también clientes con los que no contactaremos porque el modelo ha predicho que no aceptarían, pero en realidad, sí lo hubieran hecho (*False Negative* FN, negativos falsos).

Pero también, el modelo de *machine learning* acertará a efectos prácticos, esto significa que:

- Habrá clientes con los que contactaremos porque el modelo ha predicho que aceptarían y, en realidad, sí que lo hacen (True Positive TP, positivos verdaderos).

- Habrá clientes que no contactaremos porque el modelo ha predicho que no aceptarían la oferta y, en realidad, no lo hacen (*True Negative* TN, negativos verdaderos).

Para revisar esto, recordaremos un concepto aprendido anteriormente, el de la **matriz de confusión.**

En este ejemplo, contactamos a 100 clientes. 80 de ellos nos dicen que no están interesados; y 20 de ellos, que sí.

Para ello, hemos utilizado como valores la clasificación binaria:

- 0: no está interesado.
- 1: sí está interesado.

Tabla 2. Matriz de confusión con un ejemplo de *marketing*

		predicción	
		0	**1**
realidad	**0**	70	10
	1	15	5

		predicción	
		0	**1**
realidad	**0**	TN	FP
	1	FN	TP

Fuente: Martínez Hera, 2020, https://bit.ly/3MM6XI7

Medición de indicadores:

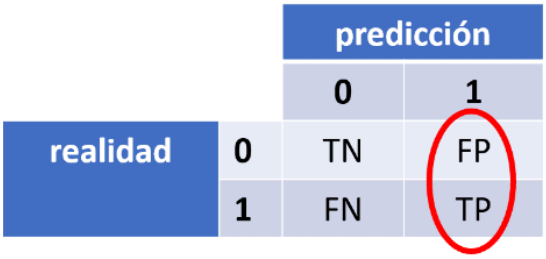
Precisión:

Con la métrica de precisión podemos medir la **calidad** del modelo de *machine learning* en tareas de clasificación. En el ejemplo, se refiere a que la precisión es la respuesta a la pregunta: ¿qué porcentaje de los clientes que contactemos estarán interesados? (Martínez Hera, 2020, https://www.iartificial.net/precision-recall-f1-accuracy-en-clasificacion/).

Para calcular la precisión, usaremos la siguiente fórmula.

Figura 27. Cálculo de precisión

$$precision = \frac{TP}{TP + FP}$$



Precisión (precision)

Fuente: Martínez Hera, 2020, https://bit.ly/3MM6XI7

En el ejemplo de *marketing*, siguiendo los datos de la matriz de confusión, tenemos que:

Figura 28. Resultado de cálculo de precisión

$$precision = \frac{TP}{TP + FP} = \frac{5}{5 + 10} = 0.33$$

Fuente: Martínez Hera, 2020, https://bit.ly/3MM6XI7

Es decir que solo un 33 % de los clientes a los que contactemos estarán realmente interesados. Esto significa que el modelo del ejemplo se equivocará un 66 % de las veces cuando prediga que un cliente va a estar interesado.

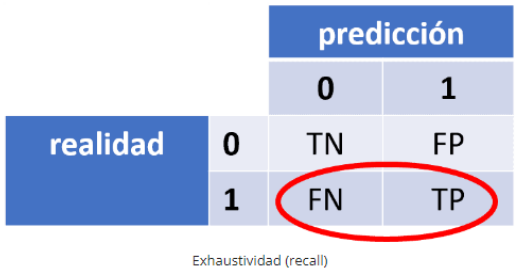
***Recall* (exhaustividad):**

La métrica de exhaustividad nos va a informar sobre la cantidad que el modelo de *machine learning* es capaz de identificar. En el ejemplo, se refiere a que la exhaustividad (*recall*) es la respuesta a la pregunta: ¿qué porcentaje de los clientes que están interesados somos capaces de identificar?

Para calcular la exhaustividad (*recall*) usaremos la siguiente fórmula. (Martínez Hera, 2020, https://www.iartificial.net/precision-recall-f1-accuracy-en-clasificacion/).

Figura 29. Cálculo de *recall*

$$recall = \frac{TP}{TP + FN}$$



Exhaustividad (recall)

Fuente: Martínez Hera, 2020, https://bit.ly/3MM6XlI7

En el ejemplo de *marketing*, siguiendo los datos de la matriz de confusión, tenemos que:

Figura 30. Resultado de cálculo de *recall*

$$recall = \frac{TP}{TP + FN} = \frac{5}{5 + 15} = 0.25$$

Fuente: Martínez Hera, 2020, https://bit.ly/3MM6XlI7

Es decir, el modelo solo es capaz de identificar un 25 % de los clientes que estarían interesados en adquirir el producto. Esto significa que el modelo del ejemplo solo es capaz de identificar 1 de cada 4 de los clientes que sí aceptarían la oferta. (Martínez Hera, 2020, https://www.iartificial.net/precision-recall-f1-accuracy-en-clasificacion/).

F1-[Score]:

El valor F1 se utiliza para combinar las medidas de precisión y *recall* en un solo valor. Esto es práctico porque hace más fácil la comparación del rendimiento combinado de la precisión y la exhaustividad entre varias soluciones.

F1 se calcula haciendo la media armónica entre la precisión y la exhaustividad. (Martínez Hera, 2020, https://www.iartificial.net/precision-recall-f1-accuracy-en-clasificacion/).

Figura 31. Cálculo de F1

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

Fuente: Martínez Hera, 2020, https://bit.ly/3MM6XlI7

En el ejemplo de *marketing*, combinando precisión y *recall* en F1 nos quedaría:

Figura 32. Resultado de cálculo de F1

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} = 2 \cdot \frac{0.33 \cdot 0.25}{0.33 + 0.25} = 0.28$$

Fuente: Martínez Hera, 2020, https://bit.ly/3MM6XlI7

El valor F1 asume que nos importa de igual forma la precisión y la exhaustividad. Esto no tiene que ser así en todos los problemas. Por ejemplo, cuando necesitamos predecir si hay riesgo de que un trozo de basura espacial se choque con un satélite, podemos valorar más la exhaustividad, a riesgo de tener una peor precisión. Por eso, elegimos F2 en lugar de F1 para esa competición de *machine learning*. (Martínez Hera, 2020, https://www.iartificial.net/precision-recall-f1-accuracy-en-clasificacion/).

Accuracy (exactitud)

La exactitud (accuracy) mide el porcentaje de casos que el modelo ha acertado. Esta es una de las métricas más usadas. El accuracy (exactitud) se calcula con la siguiente fórmula.

Figura 33. Cálculo de *accuracy*

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Fuente: Martínez Hera, 2020, https://bit.ly/3MM6XI7

La exactitud del ejemplo de *marketing* sería la siguiente.

Figura 34. Resultado de cálculo de *accuracy*

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{5 + 70}{5 + 70 + 10 + 15} = 0.75$$

Fuente: Martínez Hera, 2020, https://bit.ly/3MM6XI7

Es decir, el modelo acierta el 75 % de las veces. Como vemos, la exactitud es una métrica muy engañosa. De hecho, si tuviésemos un modelo que siempre predijera que el cliente nunca va a estar interesado, su *accuracy* sería del 80 %.

Determinar cuál es el indicador que permite medir la calidad del modelo de machine learning en tareas de clasificación. Consigna

- Accuracy.
- Precisión.
- Recall.
- F1-Score.

Justificación

Video de habilidades

Glosario

Referencias

Amat, J. (2017). Árboles de decisión, *random forest, gradient boosting* y C5.0. https://www.cienciadedatos.net/documentos/33_arboles_de_prediccion_bagging_random_forest_boosting

Amat, J. (2020). Árboles de decisión con Phyton: regresión y clasificación. https://www.cienciadedatos.net/documentos/py07_arboles_decision_python.html

[Imagen sin título sobre datos balanceados y desbalanceados]. (s. f.). https://classeval.wordpress.com/

[Imagen sin título sobre ejemplo de árbol de decisión]. (s. f.). https://becasparatodos.com/arbol-de-decisiones/

Martínez Hera, J. (2020). Precisión, *recall*, F1, *accuracy* en clasificación. https://www.iartificial.net/precision-recall-f1-accuracy-en-clasificacion/

Sanabria Castro, M. (2020). Una introducción a los árboles de decisión. https://www.grupodabia.com/post/2020-05-19-arbol-de-decision/

Wikipedia. (s. f.). Árbol de decisión. https://es.wikipedia.org/wiki/%C3%81rbol_de_decisi%C3%B3n