Trabajo de Fin de Máster

Máster universitario en Ingeniería Industrial

Aplicación de técnicas de data science para la detección de ofertas de empleo falses en Internet

MEMORIA

Autor: Alberto Mendoza Blanco
Director: Luis José Talavera Mendez

Convocatoria: Junio 2022

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

RESUMEN

En la actualidad, una de las herramientas más utilizadas por las empresas son los softwares ATS. Estos softwares facilitan la implementación de tareas monótonas o con gran posibilidad de error humano. Aun así, estos softwares están en el punto de mira de muchos estafadores que publican ofertas de trabajo fraudulentas para sus propios beneficios. Para intentar solucionar el problema, en este proyecto se plantea crear un modelo predictivo que permita detectar anuncios de trabajo fraudulentos en internet. Para ello, se seguirá la metodología CRISP-DM, la cual proporciona una descripción del ciclo de vida de este tipo de proyectos.

Durante el proyecto, se seguirán cada una de las fases de la metodología CRISP-DM. En primer lugar, se entenderá el conjunto de datos utilizados, analizando cada una de las variables tanto individualmente como en conjunto. En esta fase se observan los diferentes tipos de variables presentes, las variables textuales y no textuales. Una vez entendidos los datos, se ha realizado el procesado para que el conjunto de datos pueda ser utilizado por el modelo de regresión logística.

Para obtener un modelo capaz de realizar buenas predicciones, se han realizado una serie de pruebas. En primer lugar, con el fin de buscar que combinación de variables ofrecía mejores resultados, se han realizado pruebas con los diferentes tipos de variables del conjunto de datos original, obteniendo unas mejores predicciones con un conjunto de datos con todos los tipos de variables. En segundo lugar, al trabajar con un conjunto de datos desequilibrados, se ha planteado probar si aplicar *oversampling* ofrecía una mejora en los resultados, donde los resultados obtenidos favorecían al conjunto de datos equilibrado. En tercer lugar, para encontrar la proporción de *oversampling* que obtiene mejores resultados, se ha analizado como se comportaba el modelo para diferentes proporciones y diferentes tamaños máximos de vocabulario, obteniendo muy buenos resultados con proporciones que favorecían la clase mayoritaria y tamaños máximos de vocabulario elevados. Para acabar, se ha analizado el conjunto de datos escogido durante las pruebas para obtener los mejores resultados de este. Durante este análisis se ha visto como el modelo varía sus resultados al cambiar el valor de confianza utilizado por el modelo. Por otra parte, con el fin de reducir variables y que el modelo sea más eficiente,

se han estudiado los resultados del modelo eliminando los coeficientes con valores prácticamente nulos.

Índice

1. Introducción

1.1. Data Science

El *data science* es la rama de la ciencia que combina la estadística, métodos científicos, inteligencia artificial y análisis de datos para extraer valor de los datos. Estos datos pueden provenir de cualquier fuente que permita obtener información procesable. Hoy en día esto es sencillo gracias a los datos obtenidos a partir de smartphones, sitios web, clientes y sensores [1]. El valor que se puede obtener de los datos hace posible que el *data science* tenga un gran número de aplicaciones. Entre ellas, podemos encontrar aplicaciones en el sector de la salud, la gestión de energía, sistemas de recomendación, búsquedas y publicidad personalizadas, *gaming*, realidad aumentada, reconocimiento de imágenes y habla y, por supuesto, el reconocimiento de fraude [2].

Actualmente, una de las ramas más populares e importantes de la *data science* y la inteligencia artificial es el *machine learning.* El *machine learning* es la rama encargada de imitar la manera en la que los humanos aprendemos. Esto es posible gracias a métodos estadísticos y algoritmos que son entrenados para realizar clasificaciones o predicciones.

Podemos dividir los modelos de *machine learning* en cuatro tipos: supervisado; no supervisado; semi supervisado; y de refuerzo.

- El *machine learning* supervisado se basa en el uso de conjuntos de datos etiquetados para entrenar los algoritmos que clasifican datos o realicen predicciones. A medida que los datos de entrada se introducen en el modelo, este ajusta sus parámetros hasta adquiere la precisión deseada.

- El *machine learning* no supervisado, a diferencia del supervisado, utiliza algoritmos para analizar y clasificar datos no clasificados. El modelo es capaz de diferenciar patrones o agrupación de datos sin necesidad de intervención humana.

- El *machine learning* semi supervisado es una combinación entre los dos métodos anteriores. En este caso, se utilizan pequeños conjuntos de datos etiquetados para guiar en la clasificación y análisis de datos de un conjunto de datos sin etiquetar.

Este método soluciona el problema de no tener suficientes datos etiquetados para entrenar un modelo supervisado.

- El *machine learning* de refuerzo es un método parecido al supervisado, pero con la diferencia de que el algoritmo no se entrena con un conjunto de datos, sino que este aprende a partir del ensayo y error, donde una secuencia de resultados exitosos reforzará el modelo para adaptarse lo mejor posible a un problema en concreto [3].

Para este proyecto, se utilizará un modelo de *machine learning* supervisado. Como ya se ha comentado, el conjunto de datos utilizado se encuentra etiquetado, es decir, se conoce el resultado de la predicción para cada una de las instancias. Gracias a esto, el modelo puede calcular el error entre su predicción y el resultado real, hasta ajustarse lo mejor posible al conjunto de datos. Este proceso de aprendizaje del modelo de *machine learning* supervisado se puede dividir en tres apartados La primera parte corresponde al proceso de decisión, donde el modelo realizará una predicción o una clasificación a partir de unos datos de entrada. La segunda parte es la función de error, que como se ha explicado, tiene como objetivo evaluar la predicción realizada y definir una precisión. Por último, se produce el proceso de optimización del modelo. En el caso de que el modelo pueda ajustarse mejor a los datos de entrenamiento, los parámetros del modelo son reajustados. Este proceso se repetirá hasta que se alcance una precisión deseada.

1.2. Metodología CRISP-DM

Para llevar a cabo este proyecto, se utilizará la metodología CRISP-DM (*Cross Industry Standard Process for Data Mining*) que proporciona una descripción del ciclo de vida de un proyecto estándar de análisis de datos, detallando las fases del proyecto, las tareas a realizar y la relación entre estas, permitiendo una mejor organización del proyecto. Las diferentes fases son la comprensión del negocio, el estudio y comprensión de los datos, la preparación de los datos, el modelado, la obtención de resultados y el despliegue del modelo. Este proceso no es unidireccional, ya que la metodología permite el movimiento hacia delante y atrás. Además, el despliegue no simboliza el final del proyecto, ya que es posible descubrir nuevas interacciones con el modelo [4]. En la Figura 1 se observa un esquema del comportamiento cíclico de la metodología CRISP-DM.

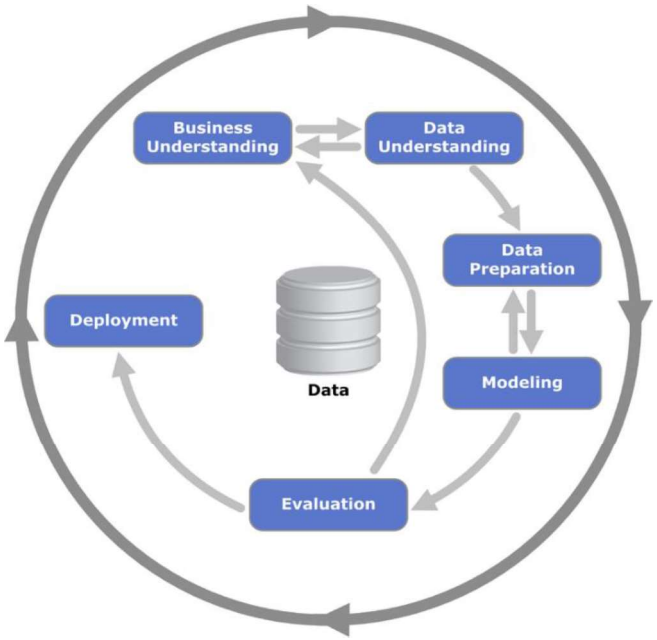

Figura 1. Esquema de las fases de la metodología CRISP [4]

A continuación, se explicarán de manera detallada las diferentes fases de la metodología CRISP-DM, indicando los objetivos de cada una y los procesos a seguir para cumplir los objetivos definidos [5].

Comprensión del negocio

La primera fase de la metodología CRISP-DM corresponde a la compresión del negocio. Esta se centra en entender y definir los objetivos del proyecto. Para ello es necesario entender lo que el cliente u organización quiere y definir los criterios para conseguir el éxito.

Durante esta fase también se definirán los recursos disponibles, los requisitos del proyecto, se determinarán los objetivos desde una perspectiva más técnica y, por último, las herramientas y tecnologías que se utilizarán en cada fase del proyecto.

Estudio y comprensión de los datos

Esta fase consiste en identificar, recoger y analizar los conjuntos de datos que pueden ayudar a cumplir los objetivos del proyecto. Las cuatro tareas principales de esta fase son: recoger los datos iniciales; examinar los datos y documentar sus propiedades; entender en profundidad los datos y las relaciones entre datos; y conocer como de limpios o sucios son nuestros datos.

Preparación de los datos

Una vez estudiados los datos, durante esta fase se prepararán los datos para la modelización. De la misma manera que en el resto de las fases, existen cinco tareas que permiten preparar los datos de manera más efectiva.

En primer lugar, será necesario seleccionar los datos que se utilizarán en el modelo y definir el porqué. Después, se realizará una limpieza de los datos y se seleccionarán los atributos que puedan ser útiles. En el caso de que tengamos datos de diferentes fuentes, es una buena idea crear un nuevo conjunto de datos con la combinación de diferentes datos. Por último, se cambiará el formato de ciertos datos para que puedan ser utilizados en el modelo.

Modelización

Durante esta fase se crearán modelos basados en diferentes técnicas. Para ello se seguirán cuatro pasos. En el primero, se determinará que técnicas de modelado se probarán. En el segundo, se realiza una primera prueba donde dependiendo del modelo es posible que sea necesario dividir los datos en datos de entrenamiento, datos de testeo y datos de validación. El siguiente paso es crear el modelo con las técnicas escogidas. Por último, es necesario interpretar los resultados del modelo y decidir si la técnica utilizada es la idónea.

Evaluación

Durante esta fase, se evaluarán los resultados obtenidos y se determinará si estos cumplen con los criterios definidos anteriormente. Con esta información se decidirá cuáles son los siguientes pasos a seguir.

Despliegue

Una vez creado el modelo y analizado sus resultados, es necesario definir como incorporar este modelo en el cliente. Para ello, se definirá un plan de despliegue y uno de monitoreo y mantenimiento del modelo para asegurar el buen funcionamiento y evitar problemas una vez el modelo haya sido incorporado.

1.3. Objetivos

En la actualidad, los procesos de contratación suceden principalmente online. Esto provoca que el número de candidatos en las diferentes ofertas de trabajo sea muy elevado, siendo necesaria la incorporación de softwares especializados encargados de facilitar la tarea [6]. Los ATS (*Applicant Tracking System*) son softwares que permiten a las empresas y organizaciones automatizar los procesos que consumen más tiempo o que más error humano pueden presentar, tales como crear los anuncios de trabajo, publicarlos, gestionar las candidaturas y comunicarse con los candidatos. Aunque estas herramientas pueden ser muy útiles, la publicación de anuncios fraudulentos por parte de estafadores puede ser un problema, que puede abarcar desde el uso de los datos para el envío de spam hasta la suplantación de identidad [7].

El objetivo de este proyecto es la creación de un modelo predictivo que permita reconocer si una oferta de trabajo es lícita o no. Para ello, se profundizará en diferentes aspectos: la metodología CRISP-DM; el entendimiento y preparación de los datos; el modelo; los resultados y su efectividad.

2. Estudio y comprensión de los datos

Una vez definidos el contexto y objetivos del proyecto, que corresponde a la primera fase de la metodología CRISP-DM, es necesario estudiar y comprender en profundidad los datos que disponemos.

El conjunto de datos que disponemos para este proyecto consta de 17.880 anuncios de trabajo, donde 17.014 corresponden a anuncios de trabajo lícitos, mientras que el resto, 866, son fraudulentos. Para cada anuncio de trabajo, encontramos información clasificada en columnas o variables, donde se encuentra distinta información expresada de varias maneras.

2.1. Análisis individual de las variables

En el conjunto de datos se encuentran 18 columnas, donde cada una contiene una información en concreto representada de diferente manera. Para saber cómo extraer la información es necesario entender perfectamente que información se encuentra en cada columna y como está expresada. Para ello se realizará un análisis en profundidad de cada variable, donde se indicará que información contiene, en que formato, los valores que puede presentar, los valores perdidos, etc.

job_id

Esta columna numera del 1 al 17.880 cada uno de los anuncios, es decir, las instancias del conjunto de datos. La información es un objeto de la clase integer y no se encuentran números perdidos.

fraudulent

fraudulent es la variable que nuestro modelo busca predecir. Esta corresponde a una variable binaria que nos indica si el anuncio es lícito o no. Los valores que puede tomar la variable son objetos de la clase integer, donde el 0 indica que el anuncio es lícito y el 1 fraudulento. Existen 17.014 anuncios que toman el valor 0 y 866 que toman el 1. En este caso no se presenta ningún número perdido.

location

La variable *location* da información sobre la localización que corresponde a la ciudad y el país. Es un objeto de la clase string. Corresponde a una variable categórica, que puede presentar 3.106 valores distintos, incluyendo los 346 valores perdidos. Debido a que el número de categorías es demasiado elevado, se ha simplificado escogiendo únicamente los dos primeros caracteres del string, que corresponden al país. Con esta simplificación, se obtienen 91 categorías diferentes, es decir, se tienen anuncios de trabajo de 91 países diferentes. Esto puede generar problemas en el modelo, ya que el vocabulario y las expresiones utilizadas pueden variar en cada país, provocando que al modelo le cueste más encontrar patrones en las variables textuales que se verán más adelante. Una solución, sería escoger únicamente un país, pero en consecuencia reducir el conjunto de datos considerablemente. En este caso, afortunadamente, la mayoría de las ofertas de trabajo son en USA, con un total de 10.593, por lo que es viable utilizar solo las ofertas de este país. Sin embargo, antes de simplificar el conjunto de datos escogiendo las ofertas en USA, se debe analizar cómo se reparten esas ofertas en la variable *fraudulent*, sobre todo las ofertas ilícitas que tienen un nombre bajo. Afortunadamente, tal y como se puede observar en la Figura 2, de las 866 ofertas de trabajo fraudulentas que hay en el conjunto de datos inicial, 725 son en USA. De las 17.014 no fraudulentas, 9.868 son en USA, obteniendo una proporción similar.

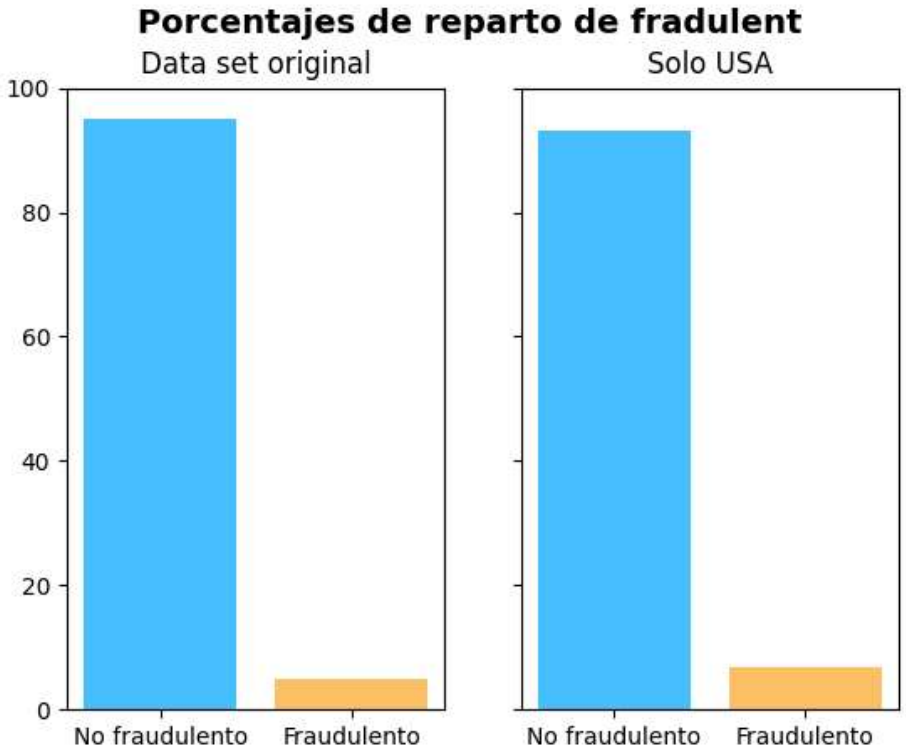

Figura 2. Comparativa del reparto de la variable fraudulent entre el conjunto de datos original y el simplificado con las ofertas de USA.

El estudio del resto de variables del conjunto de datos se realizará a partir del nuevo conjunto de datos obtenido, escogiendo únicamente las ofertas de trabajo publicadas en USA.

title

title indica el título del anuncio de trabajo. Habitualmente, el título corresponde al puesto de trabajo que se está ofertando, y en ocasiones se añade otra información como la localidad. El texto es un objeto de la clase string. En esta columna no se encuentra ningún valor perdido.

department

Indica el departamento del puesto de trabajo. La información es un objeto de la clase string. En este caso, 7.567 corresponden a valores perdidos. Corresponde a una variable

categórica con 725 posibles valores, incluyendo los números perdidos. Como el número de categorías es muy elevado, una posible solución sería analizar que departamentos son los más habituales y cuáles menos. Para los más habituales se les definiría una categoría específica, mientras que para los menos habituales una común. En nuestro caso, los 5 departamentos más habituales son los siguientes: Sales 321; Engineering 226; Marketing 176; Operations 157; y IT con 110.

salary_range

Indica el rango salarial que se ofrece por el puesto de trabajo. Es un objeto de la clase string. En la mayoría de las ofertas este valor corresponde a un número perdido, concretamente existen 9.064 números perdidos. De la misma manera que la variable *department*, *salary_range* también es una variable categórica que, en este caso, puede presentar 532 posibles valores. Para no despreciar la variable, ya que es posible que anuncios fraudulentos utilicen un buen salario para atraer a más personas, se ha decidido transformar la variable en una binaria que indica con un 0 si la oferta no indica el salario y un 1 si sí lo indica. En la Figura 3 se muestra el reparto de valores.

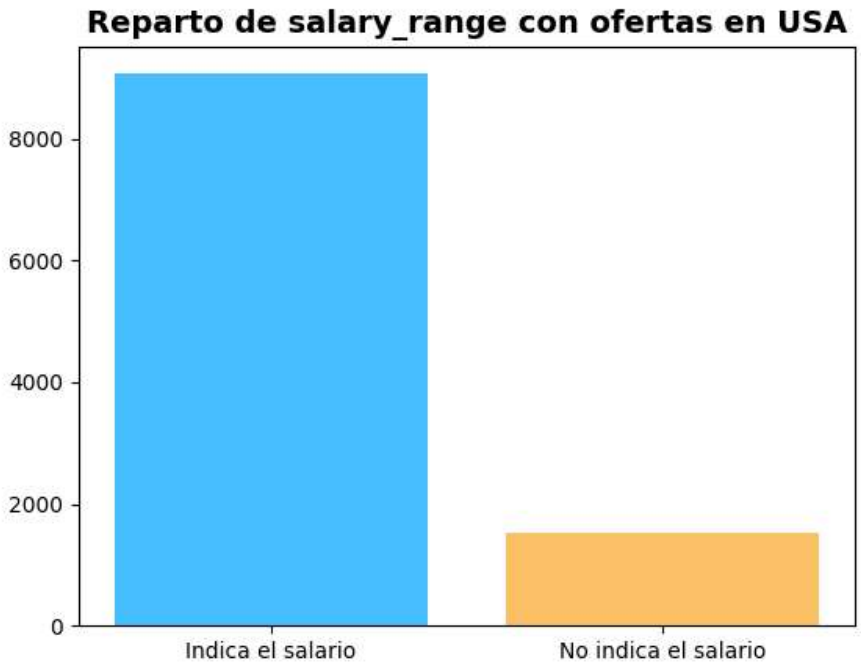

Figura 3. Reparto de la variable salary_range una vez aplicada la simplificación

company_profile

Explica el perfil profesional donde la empresa actúa. Habitualmente es un texto explicativo. La información es un objeto de la clase string. Existen 2.037 valores perdidos.

description

Texto explicativo sobre el puesto de trabajo y la empresa. Es un objeto de la clase string. En este caso no existe ningún valor perdido.

requirements

Requisitos necesarios para el puesto que se oferta. Es un objeto de la clase string. En este caso existen 1.744 valores perdidos.

benefits

Indica los beneficios que se pueden obtener del puesto de trabajo. Es un objeto de la clase string. Esta variable presenta 4.638 valores perdidos.

telecommuting

Es una variable binaria que indica si el puesto de trabajo es remoto o no. Con un 0 se indica que no lo es, mientras que con un 1 que sí. Existen 10.068 anuncios con valor 0 y 525 con valor 1. Aunque se trate de una variable booleana, esta corresponde a un objeto de la clase integer. Esta columna no presenta ningún número perdido.

has_company_logo

Se trata de una variable binaria que indica si la oferta de trabajo tiene el logo de la empresa. El 0 corresponde a la ausencia de logo, donde 14.220 anuncios presentan este

valor, mientras que el 1 indica que la oferta contiene logo con 3.660 anuncios con este valor. En este caso no hay ningún número perdido.

has_questions

Variable binaria que indica si la oferta de trabajo contiene preguntas concretas. El 0 indica que no las hay y el 1 que sí. Los valores se reparten con 2.586 con valor 0 y 8.007 con valor 1. La variable no presenta ningún número perdido.

employment_type

Indica el tipo de jornada laboral, donde esta puede ser: Other; Full-time; Part-time; Contract; y Temporary. Es un objeto de la clase string. En este caso se encuentran 3.471 valores perdidos que se podrían añadir a una nueva categoría llamada No-Definido. En este caso encontraríamos valores posibles dentro de la variable repartidos de la siguiente manera: 107 Other; 6.818 Full-time; 575 Part-time; 1.244 Contract; 95 Temporary; y 1.754 No_Definido.

required_experience

Es un objeto de la clase string que indica la experiencia necesaria para el puesto de trabajo. Los valores que pueden tomar son Intership, Not Aplicable, Mid-Senior level, Associate, Entry level, Executive y Director. También hay que tener en cuenta los 7.050 valores perdidos, que deberán tomar algún valor. De la misma manera que con la variable employment_type, es posible que la ausencia de valor de esta variable pueda ser un indicador de que el anuncio es fraudulento, por lo que será necesario crear una nueva categoría de No-Definido. Teniendo en cuenta esto, los valores se reparten de la siguiente manera: 154 Intership; 460 Not Aplicable; 2.014 Mid-Senior level; 1.405 Associate; 1.948 Entry level; 80 Executive; 265 Director; y 4.267 No_Definido.

required_education

Es un objeto de la clase string que indica la educación necesaria para el puesto de trabajo. En este caso, los valores que la variable puede presentar son 14, incluyendo los 4.460 valores perdidos. Teniendo en cuenta que una de las categorías corresponde a *Unspecified*, se ha decidido incluir los valores perdidos en esta. Con esto en mente, las 13 categorías de la variable se reparten de la siguiente manera: 3.149 Bachelor's Degree; 176 Master's Degree; 1.626 High School or equivalent; 5.174 *Unspecified*; 80 Some College Coursework Completed; 13 Vocational; 90 Certification; 218 Associate Degree; 26 Professional; 13 Doctorate; 24 Some High School Coursework; 2 Vocational – Degree; y 2 Vocational - HS Diploma.

industry

industry es una variable que toma valores correspondientes a objetos de la clase string. Esta nos indica la industria a la que pertenece la empresa de la oferta de trabajo y, al tratarse de una variable categórica, puede tomar 125 valores diferentes. En este caso, de la misma manera que la variable *required_experienc*e, los 2.672 valores perdidos se añadirán a una nueva categoría llamada No_Definido.

function

Los valores de esta variable son objetos de la variable string. Se trata de una variable categórica que nos indica el rol que se desempeñará en el trabajo. Esta puede presentar 38 valores diferentes teniendo en cuenta los 3.889 valores perdidos que se añadirán a una nueva categoría llamada No_Definido.

Durante el análisis de cada columna, en la gran mayoría se ha visto la presencia de unos valores o números perdidos. Estos valores corresponden a ofertas de trabajo, donde para la variable en cuestión no se ha añadido ningún tipo de valor, es decir, el valor es nulo. Debido a que uno de los requisitos de las bibliotecas de *machine learning* que se utilizaran

es que los datos introducidos al modelo deben presentar un valor numérico, por lo que será necesario que los valores perdidos tomen un valor. Existen varias opciones para definir un valor a estos datos, pero principalmente podemos dividirlas en tres, eliminar el anuncio de trabajo, adjudicarle un valor o eliminar la categoría. En nuestro caso solo existen números perdidos en las variables con texto o las categóricas. En el caso de las variables con texto, se ha decidido asignarles el valor de un string vacío. Para las variables categóricas, primeramente, se han añadido los valores a una variable nueva o a una ya existente. Aun así, debido a que probablemente una vez procesados los datos el número de variables sea elevado, se ha decidido eliminar las categorías No_Definido o *Unespecified* de las variables que la presentan.

Para entrenar el modelo, no es necesario utilizar todas las variables disponibles en el conjunto de datos. Es posible, que ciertas variables no aporten información o que los datos que contienen no estén completos. En nuestro caso, la variable *job_id* se ha descartado, ya que no aporta ningún tipo de información. Por otra parte, tal y como se ha comentado con la variable *department*, esta presenta un número de valores perdidos y categorías elevado. La solución que se ha propuesto no se puede llevar a cabo, ya que los departamentos más habituales tienen un nombre bajo de instancias. Es por eso, que se ha decidido no utilizar esta variable para el entrenamiento del modelo.

2.2. Relación entre variables

Una vez analizadas individualmente las columnas del conjunto de datos, estudiar la relación que tienen estas con la variable a predecir puede ser muy interesante. Dependiendo del tipo de variables con las que se quiera estudiar la relación, la correlación a utilizar varía. En este caso, debido a que las variables cuya información está en forma de texto aún no han sido procesadas, no es posible calcular la correlación de estas con la variable a predecir. El resto de las variables se han definido como variables categóricas, ya que las variables binarias se pueden entender como variables categóricas con dos posibles valores, 0 y 1. Para tratar relaciones entre variables categóricas se utilizará la correlación Chi cuadrado, concretamente el test de independencia.

Este test define como hipótesis principal que dos variables categóricas no son dependientes entre ellas, es decir, que no están relacionadas. Comparando los resultados estadísticos obtenidos en el test con unos valores críticos, se puede determinar si esta hipótesis es correcta o, en contraposición, si esta se rechaza y se acepta su hipótesis alternativa, que afirma que las variables son dependientes entre ellas.

Por ejemplo, en la Figura 4 se muestra el gráfico de la distribución de Chi cuadrado para 3 grados de libertad. Para valores de Chi cuadrado superiores al valor crítico, rechazamos nuestra hipótesis principal, ya que la probabilidad de que suceda es muy pequeña. A esta probabilidad se le conoce como *p-value* y es otra manera de aceptar o rechazar nuestra hipótesis. Para un *p-value* inferior al intervalo de confianza escogido, la hipótesis principal se rechaza y se acepta la alternativa.

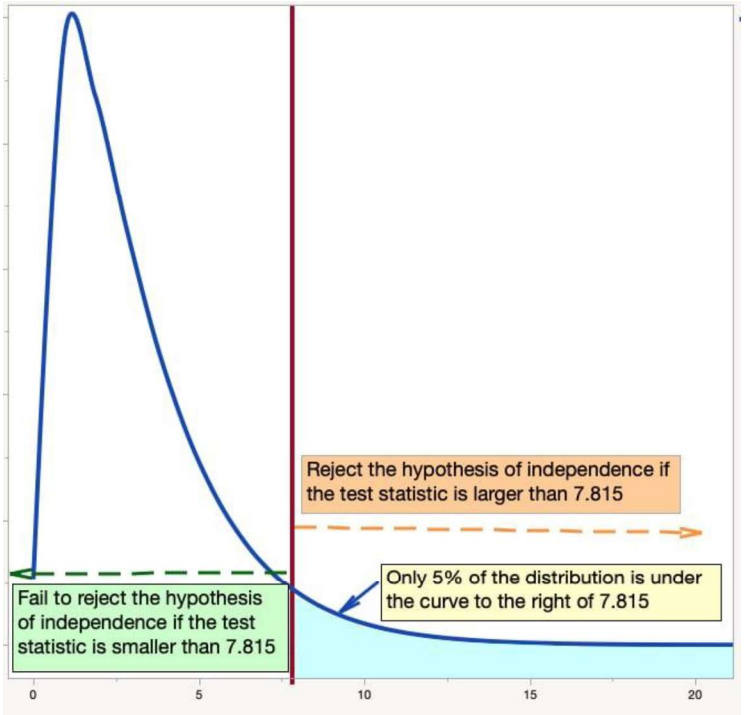

Figura 4. Gráfico de la distribución chi cuadrado para 3 grados de libertad [8]

El proceso a seguir para realizar el test de independencia de chi cuadrado es el mostrado a continuación [8]. La explicación del proceso se llevará a cabo con una sola variable categórica y la variable a predecir y, más adelante, se mostrarán los resultados obtenidos para todas las variables.

1. En primer lugar, se calculan los valores esperados a partir de la matriz de contingencia, que nos indica la repartición de los valores entre las dos variables. En la Tabla 1 se muestra la matriz de contingencia de la variable *employment_type* y *fraudulent*.

Employment Type	No Fraudulent	Fraudulent	Total fila
Contract	1.208	36	1.244
Full-Time	6.364	454	6.818
Other	95	12	107
Part-Time	542	33	575
Temporary	93	2	95
Total columna	8.302	537	**TOTAL = 8.839**

Tabla 1. Matriz de contingencia de las variables employment_type y fraudulent

Para calcular los valores esperados se multiplica el Total fila por el Total columna y se divide por el TOTAL. Con esto se obtiene la matriz de valores esperados mostrada en la Tabla 2.

Employment Type	No Fraudulent	Fraudulent	Total fila
Contract	1.168,423	75,577	1.244
Full-Time	6.403,783	414,217	6.818
Other	100,499	6,501	107
Part-Time	540,067	34,933	575
Temporary	89,228	5,772	95
Total columna	8.302	537	**TOTAL = 8.839**

Tabla 2. Matriz de valores esperados de las variables employment_type y fraudulent

Un primer análisis que nos da una idea del resultado que se obtendrá, es fijarse en la diferencia entre los valores reales y esperados. Si esta diferencia es elevada, es un indicador que las variables estudiadas son dependientes, mientras que, si la diferencia es pequeña, no lo serán.

2. El segundo paso corresponde a la propia realización del test. Para ello, calculamos la diferencia entre los valores reales y los esperados para cada combinación entre las dos variables. Después, obtenemos la diferencia cuadrática, la cual permite obtener la misma importancia para las combinaciones con más valores esperados que reales y viceversa. Por último, dividimos la diferencia cuadrática por el valor esperado y sumamos todos los valores obtenidos para cada combinación, obteniendo el valor del test chi cuadrado (Ec. 1). En la Tabla 3 se muestran los cálculos obtenidos para cada combinación.

Employment Type	**No Fraudulent**	**Fraudulent**
Contract	$Diferencia = 39{,}577$ $Dif\ cuadratica = 1.566{,}339$ $Dividio\ entre\ esperado = 1{,}341$	$Diferencia = -39{,}577$ $Dif\ cuadratica = 1.566{,}339$ $Dividio\ entre\ esperado = 20{,}725$
Full-Time	$Diferencia = -39{,}783$ $Dif\ cuadratica = 1.582{,}687$ $Dividio\ entre\ esperado = 0{,}247$	$Diferencia = 39{,}783$ $Dif\ cuadratica = 1.582{,}687$ $Dividio\ entre\ esperado = 3{,}821$
Other	$Diferencia = -5{,}499$ $Dif\ cuadratica = 30{,}239$ $Dividio\ entre\ esperado = 0{,}301$	$Diferencia = 5{,}499$ $Dif\ cuadratica = 30{,}239$ $Dividio\ entre\ esperado = 4{,}651$
Part-Time	$Diferencia = 1{,}933$ $Dif\ cuadratica = 3{,}736$ $Dividio\ entre\ esperado = 0{,}007$	$Diferencia = -1{,}933$ $Dif\ cuadratica = 3{,}736$ $Dividio\ entre\ esperado = 0{,}108$
Temporary	$Diferencia = 3{,}772$ $Dif\ cuadratica = 14{,}228$ $Dividio\ entre\ esperado = 0{,}159$	$Diferencia = -3{,}772$ $Dif\ cuadratica = 14{,}228$ $Dividio\ entre\ esperado = 2{,}465$

Tabla 3. Cálculos para cada combinación de employment_type y fraudulent

$$Chi\ cuadrado = 1{,}341 + 20{,}725 + 0{,}247 + 3{,}821 + 0{,}301 + 4{,}651 + 0{,}007 + 0{,}108 + 0{,}159 + 2{,}465 = 33{,}825 \qquad \textit{(Ec. 1)}$$

3. Por último, tal y como se ha explicado, para aceptar o rechazar nuestra hipótesis inicial, se debe comparar el valor obtenido de chi cuadrado con un valor crítico. Si el valor crítico es menor que el calculado, la hipótesis de que las variables son

independientes será correcta. En cambio, si el valor crítico es mayor que el calculado, la hipótesis es incorrecta y aceptamos la hipótesis alternativa, es decir, que las variables son dependientes.

Para obtener este valor crítico, es necesario determinar el intervalo de confianza a utilizar (α), que en este caso lo podríamos definir como el riesgo que estamos dispuestos a correr de concluir que dos variables no son independientes cuando en realidad si lo son. En nuestro caso, se ha fijado en un intervalo de confianza del 5%, es decir, $\alpha = 0{,}05$. Por otra parte, también será necesario calcular los grados de libertad entre las dos variables. Para ello se utiliza la fórmula que se muestra en la ecuación (Ec. 2), donde r y c corresponden al número de filas y columnas de la matriz de contingencia, respectivamente.

$$Grados\ de\ libertad = (r-1)\cdot(c-1) \qquad \textit{(Ec. 2)}$$

Con el intervalo de confianza y los grados de libertad, se obtiene el valor crítico de chi cuadrado. En este caso, el valor obtenido es 9,488 y, al ser este menor que el valor calculado, la hipótesis principal se rechaza y se acepta su hipótesis alternativa de que las variables son dependientes.

4. Aunque el test de independencia ya estaría acabado, es posible que ciertas variables sean dependientes, pero esta dependencia sea pequeña. Para obtener un valor que nos indique el grado de dependencia entre dos variables, se calculará el valor de V de Cramer [9]. La fórmula necesaria para calcularlo es la mostrada en la ecuación (Ec. 3), donde N corresponde al número total de observaciones y m al valor mínimo entre el número de filas - 1 y el número de columnas - 1. Por ejemplo, para la variable *employment_type*, aunque el test de independencia nos indica que hay dependencia entre ella y la variable predecir, el valor de V de Cramer es 0,083, indicando así una dependencia débil.

$$V\ de\ Cramer = \sqrt{\frac{Chi\ cuadrado}{N\cdot m}} \qquad \textit{(Ec. 3)}$$

Este proceso se ha llevado a cabo para cada una de las variables categóricas del conjunto de datos. En la Tabla 4 se muestran los resultados obtenidos para cada variable, donde encontramos el valor de Chi cuadrado calculado, su valor crítico, el *p-value* y el valor de V de Cramer.

Variable	Chi cuadrado	Valor critico	p-value	V de Cramer
salary_range	63.629	3.841	$1{,}502 \cdot 10^{-15}$	0.078
telecommuting	14.622	3.841	$1{,}313 \cdot 10^{-4}$	0.037
has_company_logo	654.087	3.841	$2{,}866 \cdot 10^{-144}$	0.248
has_questions	73.768	3.841	$8{,}787 \cdot 10^{-18}$	0.083
employment_type	33.825	9.488	$8{,}093 \cdot 10^{-7}$	0.062
required_experience	61.942	12.592	$1{,}813 \cdot 10^{-11}$	0.099
required_education	406.218	21.026	$1{,}824 \cdot 10^{-79}$	0.257
industry	1257.377	149.885	$3{,}172 \cdot 10^{-187}$	0.398
function	506,109	50,998	$2{,}712 \cdot 10^{-84}$	0,275

Tabla 4. Resultados del test de independencia de chi cuadrado para cada variable y la variable a predecir

Como se puede observar, para todas las variables, el valor de Chi cuadrado es superior a su valor crítico, por lo que todas las variables son dependientes. Aun así, según el V de Cramer, las variables más dependientes son *has_company_logo, required_education, industry* y *function*, con una dependencia moderada. El resto de las variables, al presentar un valor de V de Cramer inferior a 0,2, se consideran que tienen una dependencia débil [10].

3. Preparación de los datos

Después de estudiar y comprender los datos, el siguiente paso a seguir es preparar estos para que puedan ser utilizados en el modelo. En este caso, es necesario procesar las seis variables categóricas y las cinco variables con texto. Por otra parte, el conjunto de datos utilizado se encuentra desequilibrado, ya que existen una proporción muy pequeña de ofertas de trabajo fraudulentas en comparación a las lícitas, por lo que será necesario equilibrarlo. A continuación, se explicará en detalle el procesado de las diferentes variables y el equilibrado del conjunto de datos.

3.1. Procesamiento de lenguaje natural

Antes de profundizar en el procesamiento de las variables con texto, es necesario explicar que es el procesamiento de lenguaje natural (NLP – *Natural Language Processing*). El NLP es la rama de la inteligencia artificial centrada en dar la habilidad a las máquinas de entender el texto hablado y escrito de la misma manera que los humanos somos capaces, combinando varias disciplinas como la estadística y el *machine learning* [11].

Para realizar el procesado del texto se ha utilizado *nltk*, una librería que permite realizar el procesamiento de lenguaje natural. El procesamiento consta de dos partes: la primera corresponde al procesamiento propiamente dicho, donde ser realizarán una serie de procesos que permitirán extraer la información del texto; y la segunda, corresponde al vectorizado que es necesario para darle un valor numérico y, que así, pueda ser utilizado en el modelo. En nuestro caso, para extraer la información de los textos de las variables textuales, se definirá un vocabulario para cada una de ellas. Este vocabulario corresponde a la agrupación de tokens o palabras que permite representar los textos de cada variable. Una vez definido el vocabulario, se le adjudicará un valor a cada uno de los tokens que lo conforman.

3.1.1. Procesado del texto

Para el procesamiento del texto, se han seguido una serie de pasos. En primer lugar, se han pasado todas las palabras a minúsculas, ya que esto hará que el vocabulario final se reduzca y la información obtenida sea la misma. Una vez convertido el texto a minúsculas, se ha tokenizado. La tokenización corresponde a la separación de las palabras del texto en tokens, en nuestro caso, individuales. La tokenización, a diferencia de otras técnicas, permiten separar las palabras del texto teniendo en cuenta la puntuación y otros factores. La tokenización es el primer paso para poder realizar operaciones que permitirán simplificar el texto y extraer la máxima información. Como la tokenización permite separar los signos de puntuación de las palabras e identificar cada uno como tokens independientes. Esto permite poder eliminarlos de manera sencilla, ya que estos no aportan ningún tipo de información al texto y permite reducir el vocabulario final. Supongamos el texto de la Figura 5, se aplica en primer lugar la tokenización, donde cada token corresponde a una palabra del texto. Como se puede observar, la tokenización permite separar las palabras y los signos de puntuación del texto. Como estos últimos no aportan información, los eliminamos.

Text = "NLP combines computational linguistics, rule-based modelling of human language, with statistical, machine learning, and deep learning models. Together, these technologies enable computers to process human language."

Tokenización = ['nlp', 'combines', 'computational', 'linguistics', ',', 'rule-based', 'modeling', 'of', 'human', 'language', ',', 'with', 'statistical', ',', 'machine', 'learning', ',', 'and', 'deep', 'learning', 'models', '.', 'together', ',', 'these', 'technologies', 'enable', 'computers', 'to', 'process', 'human', 'language', '.']

Eliminación signos de puntuación = ['nlp', 'combines', 'computational', 'linguistics', 'rule-based', 'modelling', 'of', 'human', 'language', 'with', 'statistical', 'machine', 'learning', 'and', 'deep', 'learning', 'models', 'together', 'these', 'technologies', 'enable', 'computers', 'to', 'process', 'human', 'language']

Figura 5. Tokenización y eliminación de los signos de puntuación del texto

Con el texto tokenizado y con los signos de puntuación eliminados, el siguiente paso es eliminar las palabras que no aportan significado, es decir, las *stopwords.* Las *stopwords*, o palabras vacías, corresponden a las palabras que por sí solas no aportan ningún significado, sino que modifican o acompañan a otras. Este grupo suele estar formado por artículos, pronombres, preposiciones, adverbios e incluso algunos verbos. En la Figura 6, se puede observar la lista de tokens obtenidos una vez eliminadas las *stopwords*.

Ejemplos de *Stopwords* = ['more', 'what', "should've", 'is', "you'd", "it's", 'both', 're', 'wouldn', 'there', 'don', 'weren', "isn't", 'yourself', 'why', 'yourselves', 'through', 'her', 'of', "hadn't", 's', 'isn', 'himself', "hasn't", 'themselves', 'their', 'just', 'if', 'this', 'then', 'will', 'which', 'mustn', 'hers', 'own', 'as', 'o', 'ain', "shan't", 'only', "you've", 'those', 'from', 'about', 'had', 'here', 'out', 'or', "don't", 'doing', "aren't", 'during', 'can', 'that', 'against', 'same', 'does', 'how', 'now', 'myself', "mustn't", 'itself', 'should', 'off', 'in', 'once', 've', 'a', 'whom', "wouldn't", 'yours']

Eliminación *Stopwords* = ['nlp', 'combines', 'computational', 'linguistics', 'rule-based', 'modelling', 'human', 'language', 'statistical', 'machine', 'learning', 'deep', 'learning', 'models', 'together', 'technologies', 'enable', 'computers', 'process', 'human', 'language']

Figura 6. Ejemplos de stopwords y su eliminación de la tokenización obtenida en la Figura 5

Por último, para reducir aún más el vocabulario final, se realizará la lematización de los tokens restantes. La lematización es un proceso que consiste en, dada una forma flexionada, hallar el lema correspondiente, entendiendo como forma flexionada a las palabras con morfemas flexivos, es decir, que pueden variar en género, número, persona, tiempo, etc. El lema, en cambio, es la forma que se acepta como representante de todas las formas flexionadas de una misma palabra. En la Figura 7 se muestra la lematización de los tokens obtenidos anteriormente.

Lematización = ['nlp', 'combine', 'computational', 'linguistics', 'rule-based', 'model', 'human', 'language', 'statistical', 'machine', 'learn', 'deep', 'learn', 'model', 'together', 'technology', 'enable', 'computer', 'process', 'human', 'language']

Figura 7. Lematización de los tokens de la Figura 6.

Para poder aplicarlo a cada una de las instancias de las variables con texto, se ha creado una función llamada *clean_text*, que tiene como parámetro *text* que corresponde al texto que se quiera procesar.

3.1.2. Vectorización

Seguidamente, después de crear la función que nos permite procesar el texto, es necesario vectorizarlo. Se entiende por vectorización a la codificación del texto para asignarle un valor numérico. Existen varios métodos de vectorización, pero en nuestro caso utilizaremos la vectorización TF-IDF (*Term Frequency-Inverse Document Frequency*), principalmente, por ser un algoritmo capaz de dar más peso a los términos más importantes en función de su frecuencia en el texto e instancias y por ser ampliamente utilizado en este tipo de aplicaciones [12].

TF-IDF combina dos conceptos, la frecuencia del término (TF) y la frecuencia del documento (IDF). La frecuencia del término es el número de veces que un término aparece en el texto e indica como de importante es este en el texto. La frecuencia del documento indica el número de documentos que contienen el término en cuestión, indicando lo común que es este. En este caso, la inversa de la frecuencia del texto (IDF) busca reducir el peso de un término si este es muy común en los documentos. Este algoritmo busca dar prioridad a los términos que son importantes en el texto, pero que no se repiten en los diferentes documentos para así encontrar términos que permitan distinguir los diferentes documentos entre ellos [12]. Este algoritmo debe aplicarse a cada una de las variables textuales por separado, creando un vocabulario con diferentes pesos para cada una de ellas.

Para obtener el peso de un término se puede calcular con la ecuación (Ec. 4), donde $w_{i,j}$ es el peso del término i en el documento j, $t_{i,j}$ es la frecuencia del término i en el documento j y idf_i corresponde al peso IDF.

$$w_{i,j} = tf_{i,j} \cdot idf_i \qquad \textit{(Ec. 4)}$$

Para calcular el peso de IDF necesario en la ecuación anterior, se utiliza la ecuación (Ec. 5), donde idf_i es el peso para el término i, df_i es el número de documentos que contienen el término i y n corresponde al número de documentos.

$$idf_i = log\left(\frac{n}{df_i}\right) \qquad \textit{(Ec. 5)}$$

Para implementar el algoritmo de vectorización TF-IDF en *Python* se ha creado el vectorizador indicando una serie de parámetros.

Con el parámetro *min_df* se ha definido una frecuencia mínima del token entre documentos (DF). Esto se ha hecho para evitar que en el vocabulario aparezcan tokens muy poco comunes que no aportan información, como por ejemplo enlaces de páginas web. Para evitar vocabularios muy grandes y analizar los resultados para tamaños de vocabulario diferentes, se ha utilizado el parámetro *max_features*, que nos indica el número máximo de tokens que habrá por vocabulario. Para decidir que tokens se utilizarán, estos se ordenan según la frecuencia del término (TF). También se ha definido el rango de tamaño de los *ngrams*, que corresponde a la agrupación de tokens. En nuestro caso, se ha fijado para que únicamente defina *unigrams*, es decir, tokens individuales. El parámetro *lowercase*, que por defecto tiene el valor *True*, se ha definido *False* para que no pase todos los términos a minúsculas, ya que se utilizará la función explicada en el apartado 3.1.1 que ya lo implementa. Por último, si no se define ninguna función de procesado de texto, el vectorizador tokenizará el texto. Aun así, la librería utilizada permite un mejor tokenizado, por lo que en el parámetro *tokenizer* se ha asignado la función *clean_text*.

Una vez definido el vectorizador, se ha ajustado y transformado la columna indicada creando un vocabulario con unos pesos para cada instancia del conjunto de datos. Este proceso se ha repetido para todas las variables textuales y se ha unificado todo en un único

conjunto de datos, donde las columnas corresponden a los diferentes términos del vocabulario y los valores a los pesos calculados con la ecuación (Ec. 4).

En la Figura 8 se muestra el proceso a seguir para vectorizar tres textos [12]. Como se puede observar, el primer paso se basa en crear una matriz de frecuencia de los términos de cada texto. En segundo lugar, se calcula la inversa de la frecuencia del documento, es decir, la frecuencia de cada término en los tres textos. Por último, se calculan los pesos de los términos con la (Ec. 4).

Texto 1: i love natural language processing but i hate python

Texto 2: i like image processing

Texto 3: i like signal processing and image processing

	and	but	hate	i	image	language	like	love	natural	processing	python	signal
Text 1	0	1	1	2	0	1	0	1	1	1	1	0
Text 2	0	0	0	1	1	0	1	0	0	1	0	0
Text 3	1	0	0	1	1	0	1	0	0	2	0	1

Term	and	but	hate	i	image	language	like	love	natural	processing	python	signal
IDF	0.47712	0.47712	0.4771	0	0.1760913	0.477121	0.1760913	0.477121	0.47712125	0	0.477121	0.477121

	and	but	hate	i	image	language	like	love	natural	processing	python	signal
Text 1	0	0.47712	0.4771	0	0	0.477121	0	0.477121	0.47712125	0	0.477121	0
Text 2	0	0	0	0	0.1760913	0	0.1760913	0	0	0	0	0
Text 3	0.47712	0	0	0	0.1760913	0	0.1760913	0	0	0	0	0.477121

Figura 8. Ejemplo de vectorización de tres textos [12]

3.2. Procesamiento de las variables categóricas

El otro tipo de variable que se encuentra en el conjunto de datos son las variables categóricas, que presentan un número concreto de valores únicos o categorías. De nuevo, debido a que uno de los requisitos de las bibliotecas de *machine learning* utilizadas durante el proyecto es que los valores de las variables deben ser numéricos, cuando los valores de las variables categóricas no son lo son, es necesario procesarlas y darles un valor. En nuestro caso, ciertas variables categóricas presentan varios valores en forma de string y, por lo tanto, hay que procesarlas. Para procesarlas, se convertirán a lo que se conoce como *dummy variables*. Las *dummy variables* son variables binarias, es decir, variables que solo presentan dos valores, 0 o 1. Para poder transformar una variable categórica con varias clases, se añaden tantas columnas como clases y se le adjudica un valor binario. Esto implica que el número de columnas del conjunto de datos aumenten considerablemente.

Para el procesado, se ha utilizado un método de la librería pandas, *get_dummies*, donde se introduce el conjunto de datos inicial y las columnas donde se quiere aplicar. Este método crea un conjunto de datos nuevo, eliminando la variable indicada y añadiendo tantas variables como categorías tenía la variable inicial, adjudicándoles un valor binario. Cabe destacar que es necesario que las columnas donde se quiere aplicar la función no tengan valores perdidos, por lo que es necesario adjudicarles un valor antes de aplicar dicha función. Para ello se ha creado la función *nan_to_cat*, que tiene como parámetros un texto, que puede ser nulo o no, y la categoría que se le quiere adjudicar al valor nulo.

Al aplicar este proceso, el número de variables final del conjunto de datos se dispara. Para reducir el número de variables y afectar lo mínimo a la información que el conjunto de datos contiene, se han eliminado las variables No_Definido y *Unspecified*. Aquellas instancias con un 1 en estas variables, una vez eliminadas, pasarán a tener valor de 0 en el resto de las categorías de la variable principal, por lo que la información que dan es la misma.

En la Tabla 5 se muestra un ejemplo didáctico de la binarización de la variable *employment_type*, donde se han añadido tantas columnas como categorías tenía la variable y, por último, se ha eliminado la columna No_Definido. Para este caso, se eliminaría la categoría *employment_type* y se añadirían 5 nuevas variables pertenecientes a sus categorías.

Instancia	**employment_type**
1	Full Time
2	Part Time
3	Full Time
4	Other
5	Contract
6	No_Definido
7	No_Definido
8	Temporary
9	Other
10	Temporary

Other	**Full Time**	**Part Time**	**Contract**	**Temporary**
0	1	0	0	0
0	0	1	0	0
0	1	0	0	0
1	0	0	0	0
0	0	0	1	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	1
1	0	0	0	0
0	0	0	0	1

Tabla 5. Ejemplo de binarización de la variable employment_type

4. Marco teórico

4.1. Modelo. Regresión logística

4.1.1. Regresión logística

La regresión logística es un algoritmo estadístico utilizado para la predicción de variables binarias. Este utiliza la ecuación logística (Ec. 6) como representación, donde los valores de entrada x_i, son combinados linealmente utilizando coeficientes para predecir el valor de salida y_i. El coeficiente β_0 se conoce como el término de intersección y β_1 corresponde a los coeficientes para la variable input. Estos coeficientes son los que deberán ser aprendidos con el conjunto de datos de entrenamiento para obtener el modelo predictivo [13]. Cabe destacar, que las ecuaciones mostradas a continuación, con el fin de simplificar la explicación, corresponden a un modelo con una sola entrada. En el caso de tener un modelo con múltiples variables, se tendrá un coeficiente β_0 y tantos coeficientes como variables de entrada.

$$y = \frac{e^{\beta_0+\beta_1 \cdot x_i}}{1 + e^{\beta_0+\beta_1 \cdot x_i}} \qquad \textit{(Ec. 6)}$$

Para poder clasificar la variable binaria, la regresión logística calcula la probabilidad de la primera clase, que corresponde a la clase con valor 1 (Ec. 7). Esta probabilidad es después transformada a valores binarios a partir de un valor de confianza.

$$P(x_i) = P(y = 1|x_i) \qquad \textit{(Ec. 7)}$$

La regresión logística es un método lineal, pero las predicciones son transformadas utilizando la función logística, por lo que estas no pueden entenderse como combinaciones lineales de los valores de entrada. Con la ecuación (Ec. 7) y la función logística, se obtiene la ecuación del modelo (Ec. 8).

$$P(x_i) = \frac{e^{\beta_0+\beta_1 \cdot x_i}}{1 + e^{\beta_0+\beta_1 \cdot x_i}} \qquad \textit{(Ec. 8)}$$

Para entender mejor la linealidad del modelo, se deben reorganizar las ecuaciones obtenidas. A partir de la ecuación del modelo, se obtiene la ecuación (Ec. 9), donde se puede observar que el término de la izquierda se comporta linealmente. Este término corresponde al logaritmo de la probabilidad de la primera clase y es conocido como las posibilidades de la clase principal. Estas posibilidades o *odds* son calculadas como la probabilidad del evento, en este caso de que suceda la clase principal, entre la probabilidad de que no suceda el evento (Ec. 10). Con esto en mente, podemos escribir la función (Ec. 9) como la mostrada en la ecuación (Ec. 11), donde se relaciona el logaritmo de la posibilidad de la primera clase, conocido como *log-odds* o *probit,* de manera lineal.

$$\ln\left(\frac{P(x_i)}{1-P(x_i)}\right) = \beta_0 + \beta_1 \cdot x_i \qquad \textit{(Ec. 9)}$$

$$odds = \frac{P(x_i)}{1-P(x_i)} \qquad \textit{(Ec. 10)}$$

$$\ln(odds) = \beta_0 + \beta_1 \cdot x_i \qquad \textit{(Ec. 11)}$$

Por último, con la ecuación (Ec. 11) se aísla la posibilidad obteniendo la ecuación (Ec. 12). Todo este proceso, ayuda a entender la linealidad del *log-odds* de la clase principal respecto a los valores de entrada.

$$odds = e^{\beta_0 + \beta_1 \cdot x_i} \qquad \textit{(Ec. 12)}$$

4.1.2. Estimador *maximum-likelihood*

Una vez entendido en que se basa el modelo y obtenida la ecuación de este (Ec. 8), es necesario obtener los coeficientes a partir de los datos de entreno. Para ello, se utiliza la estimación *maximum-likelihood*. Esta estimación busca los mejores coeficientes que permitan predecir con un valor muy cercano a 1 una instancia de la clase principal y con un valor muy cercano a 0 para la otra clase. Para lograrlo, se realiza un proceso iterativo que busca minimizar el error entre la probabilidad obtenida por el modelo y la que debería ser para el dato en concreto. Por ejemplo, para una instancia de la clase principal, el modelo buscará minimizar el error entre la probabilidad obtenida por el modelo y 1 [13]. De la misma manera que con las ecuaciones del modelo de regresión logística, con el fin de simplificar las fórmulas comentadas, estas corresponden a un modelo con una sola

entrada. En el caso de tener un modelo con más variables de entrada, se deberá estimar el coeficiente β_0 y tantos coeficientes como variables de entrada haya.

Para entender el proceso que engloba la estimación de coeficiente con el estimador *maximum-likelihood* y el proceso iterativo de Newton Raphson, se parte de la ecuación (Ec. 13), donde y_i es una variable aleatoria independiente del resto de variables que sigue una distribución de Bernoulli, $Y_i \sim Bernoulli\ (P(x_i))$ [14].

$$p(y_1, y_2, \dots y_n) = \prod_{i=1}^{n} P(x_i)^{y_i} \cdot (1 - P(x_i))^{1-y_i} \qquad \textit{(Ec. 13)}$$

Con la ecuación del modelo (Ec. 8), la anterior ecuación se puede representar tal y como se muestra en la ecuación (Ec. 14), obteniendo la función *likelihood*. Para simplificar la derivada de la función, se utiliza la función *log-likelihood* mostrada en la ecuación (Ec. 15).

$$\begin{aligned} p(y_i, y_2, \dots y_n) &= L(\beta_0, \beta_1) \\ &= \prod_{i=1}^{n} y_i \cdot \left(e^{\beta_0 + \beta_1 \cdot x_i}\right)^{y_i} \cdot \frac{1}{1 + e^{\beta_0 + \beta_1 \cdot x_i}} \end{aligned} \qquad \textit{(Ec. 14)}$$

$$\begin{aligned} l(\beta_0, \beta_1) &= \ln\left(L(\beta_0, \beta_1)\right) \\ &= \sum_{i=1}^{n} y_i \cdot (\beta_0 + \beta_1 \cdot x_i) - \sum_{i=1}^{n} ln(\, e^{\beta_0 + \beta_1 \cdot x_i}) \end{aligned} \qquad \textit{(Ec. 15)}$$

A partir de aquí, aplicando las derivadas respecto β_0 y β_1, se obtienen las ecuaciones (Ec. 16 y 17), respectivamente.

$$\frac{\delta l(\beta_0, \beta_1)}{\delta \beta_0} = \sum_{i=1}^{n} y_i - \frac{e^{\beta_0 + \beta_1 \cdot x_i}}{1 + e^{\beta_0 + \beta_1 \cdot x_i}} \qquad \textit{(Ec. 16)}$$

$$\frac{\delta l(\beta_0, \beta_1)}{\delta \beta_1} = \sum_{i=1}^{n} y_i x_i - x_i \frac{e^{\beta_0 + \beta_1 \cdot x_i}}{1 + e^{\beta_0 + \beta_1 \cdot x_i}} \qquad \textit{(Ec. 17)}$$

De nuevo, con la ecuación del modelo (Ec. 8), se obtienen las siguientes simplificaciones (Ec. 18 y 19). Debido a que extraer los parámetros β_0 y β_1 puede ser difícil, se utiliza el método iterativo de Newton Raphson para obtener los valores estimados $\hat{\beta}_0$ y $\hat{\beta}_1$.

$$\frac{\delta l(\beta_0, \beta_1)}{\delta \beta_0} = \sum_{i=1}^{n} (y_i - P(x_i)) = 0 \qquad \textit{(Ec. 18)}$$

$$\frac{\delta l(\beta_0, \beta_1)}{\delta \beta_1} = \sum_{i=1}^{n} (y_i x_i - x_i\, P(x_i)) = \sum_{i=1}^{n} (y_i - P(x_i)) x_i = 0 \qquad \textit{(Ec. 19)}$$

Para estimar los valores de los coeficientes, se siguen los siguientes pasos:

1. Obtención de un valor inicial estimado de los parámetros.
2. Obtención de los parámetros de la iteración (k+1) (Ec. 20)

$$\beta_{(k+1)} = \beta_{(k)} - I(\beta)^{-1}_{(k)} \cdot U(\beta)_{(k)} \qquad \textit{(Ec. 20)}$$

3. El método iterativo continua hasta que los valores obtenidos en la nueva iteración son prácticamente iguales que los de la anterior, $\beta_{(k+1)} \approx \beta_{(k)}$.

Los valores de $I(\beta)$ y $U(\beta)$ se calculan con las ecuaciones (Ec. 21 y 22), respectivamente.

$$I(\beta) = \begin{bmatrix} \frac{\delta^2 l(\beta_0, \beta_1)}{\delta {\beta_0}^2} & \frac{\delta^2 l(\beta_0, \beta_1)}{\delta \beta_0 \delta \beta_1} \\ \frac{\delta^2 l(\beta_0, \beta_1)}{\delta {\beta_1}^2} & \frac{\delta^2 l(\beta_0, \beta_1)}{\delta \beta_1 \delta \beta_0} \end{bmatrix}^{-1}$$

$$I(\beta) = -\begin{bmatrix} \sum_{i=1}^{n} P(x_i)(1 - P(x_i)) & \sum_{i=1}^{n} x_i P(x_i)(1 - P(x_i)) \\ \sum_{i=1}^{n} x_i P(x_i)(1 - P(x_i)) & \sum_{i=1}^{n} x_i^2 P(x_i)(1 - P(x_i)) \end{bmatrix}^{-1} \qquad \textit{(Ec. 21)}$$

$$U(\beta) = \begin{bmatrix} \frac{\delta l(\beta)}{\delta \beta_0} \\ \frac{\delta l(\beta)}{\delta \beta_0} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i - P(x_i) \\ \sum_{i=1}^{n} y_i x_i - x_i P(x_i) \end{bmatrix} \qquad \textit{(Ec. 22)}$$

4.1.3. Predicciones con regresión logística

Para obtener predicciones con la regresión logística, simplemente se deberán introducir los números de los valores de entradas para así obtener una probabilidad. Una vez obtenida esa probabilidad, fijando un valor de confianza, si la probabilidad se encuentra por encima de ese valor, se clasificará esa instancia en la clase principal. Si, por lo contrario, la probabilidad obtenida se encuentra por debajo del valor de confianza fijado, se clasificará en la otra clase [13].

Habitualmente, ese valor de confianza se fija en 0,5. Aun así, se puede jugar con ese valor para obtener unas mejores predicciones. Si reducimos este valor de confianza, el modelo clasificará más instancias como la clase principal, pero se corre el riesgo de clasificarlas erróneamente. En cambio, si aumentamos el valor de confianza, el modelo clasificará menos instancias como la clase principal, pero tendrá un riesgo menor de clasificarlas de manera incorrecta.

4.2. Separación entrenamiento – test

Separar el conjunto de datos en uno de entrenamiento y otro de test, es un proceso muy habitual en algoritmos de *machine learning*, que permite obtener de manera sencilla una idea de cómo el algoritmo predice.

El proceso consiste en utilizar el conjunto de datos original para obtener dos subconjuntos. El primero, llamado conjunto de datos de entrenamiento, se utilizará para ajustar el modelo. El segundo, en cambio, no es utilizado para entrenar el modelo, sino que, con los *inputs* se realizan predicciones con el modelo y se comparan con los valores reales. Este se le conoce como conjunto de datos de test y permite estimar el rendimiento que tendrá el modelo. En la Figura 9, se muestra un gráfico que permite entender mejor en que se basa la separación del conjunto de datos.

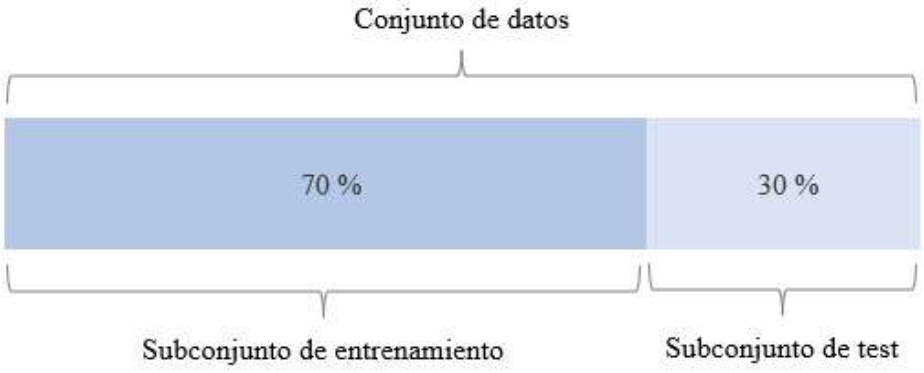

Figura 9. División del conjunto de datos en subconjunto de entreno y test

Es importante realizar esta separación, ya que, si se utilizase el mismo conjunto de datos para entrenar y evaluar el modelo, al utilizar instancias iguales para los dos procesos, el modelo obtendría un rendimiento muy optimista que no sería representativo de la realidad.

Una vez entendido porque es importante y necesario realizar la separación entre entrenamiento y test, se hablará sobre las proporciones de cada uno de los subconjuntos. Estas proporciones varían según las necesidades del problema, tales como el coste computacional para entrenar o evaluar el modelo, o la representatividad del set de entreno o test. Aun así, las proporciones más típicas son 80-20, 70-30 o 50-50.

Por último, al momento de implementar esta separación, hay que tener en cuenta que esta escoge de manera aleatoria las instancias de entreno y test. Para poder comparar los resultados de diferentes evaluaciones, es necesario utilizar siempre los mismos subconjuntos, por lo que se deberá utilizar una semilla [15].

4.3. Métricas

Para evaluar los resultados obtenidos en el modelo, es necesario realizar un análisis estadístico de este. Para ello, se utilizarán una serie de métricas y herramientas que permiten entender los resultados obtenidos del modelo. Estas son la matriz de confusión, la *Precision*, el *Recall* y el F1.

4.3.1. Matriz de confusión

La matriz de confusión es una herramienta que permite visualizar el desempeño de un algoritmo de aprendizaje supervisado. Cada columna de la matriz representa el número de predicciones de cada clase y cada fila representa las instancias de la clase real. En la Figura 10, se muestra un ejemplo de matriz de confusión binaria, donde se entiende como valor positivo el 1 y como valor negativo el 0 [16].

		Predicciones	
		Fraudulento (1)	**No Fraudulento (0)**
Valores Reales	**Fraudulento (1)**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	No Fraudulento (0)	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Figura 10. Matriz de confusión binaria

Como se puede observar en la matriz de confusión, existen cuatro resultados posibles [16].

- Verdadero positivo: El valor real es positivo y la predicción realizada por el modelo también es positiva. En nuestro caso, el anuncio de trabajo es fraudulento y el modelo también lo predijo así.

- Verdadero negativo: El valor real es negativo y la predicción del modelo también lo es. En estas situaciones, el anuncio de trabajo era lícito y el modelo lo predijo así.

- Falso positivo: Sucede cuando el valor real es negativo y el modelo lo predijo positivo. Corresponde a un anuncio lícito, pero con una predicción positiva. Este tipo de error se conoce en estadística como error tipo I.

- Falso negativo: El valor real es positivo, pero el modelo lo predijo como negativo. Corresponde a un anuncio fraudulento que el modelo predijo como lícito. Este error se conoce en estadística como error tipo II.

Con los cuatro resultados posibles surgen una serie de métricas que nos dan información sobre estos tipos de aciertos y errores que el modelo ha cometido con el conjunto de datos de test. Entre ellas encontramos la *Precision*, el *Recall* y el F1, que son la que se utilizaran para evaluar los resultados del modelo.

4.3.2. *Precision*

La precisión se relaciona como la dispersión de un conjunto de valores obtenidos a partir de repetidas mediciones de una magnitud, por lo que a menor dispersión se obtendrá una mayor precisión. En nuestro caso, se puede entender como precisión, a la proporción de casos positivos o negativos detectados correctamente. Con las ecuaciones (Ec. 23 y 24) se pueden obtener los valores de *Precision* para las dos clases. Como se puede observar, esta métrica nos dará información sobre lo bien que nuestro modelo clasifica los valores positivos o negativos de la clase [16].

$$Precision\ (Fraudulento) = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Positivos} \quad \textit{(Ec. 23)}$$

$$Precision\ (No\ Fraudulento) = \frac{Verdaderos\ Negativos}{Verdaderos\ Negativos + Falsos\ Negativos} \quad \textit{(Ec. 24)}$$

4.3.3. *Recall*

El *Recall* o sensibilidad indica la capacidad del modelo de discriminar casos positivos de negativos. Esta métrica, de la misma manera que la *Precision*, se puede calcular tanto para valores positivos como negativos de la clase objetivo. Para los valores positivos, el *Recall* indica la tasa de verdaderos positivos, es decir, la proporción de casos positivos que fueron correctamente identificados por el algoritmo. Para valores negativos, en cambio, el *Recall* indica la tasa de verdaderos negativos, es decir, los casos negativos que el modelo ha clasificado correctamente. Este valor muestra lo bueno que es el algoritmo en detectar esa clase. Para poder calcular los valores de estas métricas se utilizan las ecuaciones (Ec. 25 y 26) [16].

$$Recall\ (Fraudulento) = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Negativos} \qquad (Ec.\ 25)$$

$$Recall\ (No\ Fraudulento) = \frac{Verdaderos\ Negativos}{Verdaderos\ Negativos + Falsos\ Positivos} \qquad (Ec.\ 26)$$

4.3.4. F1

En situaciones donde distribución entre valores positios y negativos en una clase se encuentra desbalanceada, la métrica F1 gana más peso. Esta métrica resume la *Precision* y el *Recall* en un solo valor. Su valor se calcula según la ecuación (Ec. 27). De la misma manera que las otras métricas, la F1 se puede calcular tanto para valores positivos como negativos [16].

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \qquad (Ec.\ 27)$$

4.4. *Oversampling*

Se entiende como un conjunto de datos no equilibrado aquel en que la variable a predecir tiene más observaciones en una clase que en otra. Cuando se entrena un modelo con un conjunto de datos no equilibrado, el algoritmo tiende a categorizar la clase con más instancias, dando la falsa sensación de que se tiene un modelo muy preciso. En los momentos en que el modelo debe predecir situaciones no habituales, este priorizará la clase con más instancias. Aun así, a causa de la falsa precisión del modelo, estas falsas detecciones quedan escondidas [17].

Para poder solucionar este problema, existen dos opciones, el *undersampling* y el *oversampling*.

- *Undersampling*: La idea del *undersampling* es reducir la ratio de las instancias de las clases mayoritarias y minoritarias. Para ello se puede seleccionar de manera aleatoria instancias de las dos clases siguiendo una ratio más equilibrada, 50/50 o 60/40.

- *Oversampling*: Para el *oversampling*, se busca crear instancias sintéticas de la clase minoritaria a partir de los datos disponibles. El principal beneficio que existe del *oversampling* es que a la hora de entrenar se podrán utilizar más datos, que podrán llevar a mejores resultados. Existen varios algoritmos que permiten esto, entre ellos encontramos: *Random OverSampler*; VAE (*Variational Autoencoders*); SMOTE (*Synthetic Minority Over-sampling Technique*); o MSMOTE (*Modified Synthetic Minority Over-sampling Technique*).

Habitualmente, para no perder información y aumentar la cantidad de datos, se suelen utilizar técnicas de *oversampling.* El *Random OverSampling* es uno de los métodos más sencillos de *oversampling*. Este se basa en duplicar de manera aleatoria las instancias de la clase minoritaria. Esto quiere decir que ciertas instancias escogidas por el método para añadirlo al nuevo conjunto de datos más balanceado pueden aparecer varias veces. Esta técnica puede ser efectiva en situaciones donde se utilizan modelos que son afectados por distribuciones sesgadas o que utilizan procesos iterativos para aprender sus coeficientes

[18]. En la Figura 11 se observa la diferencia en la clasificación una vez aplicado el *Random Oversampling*.

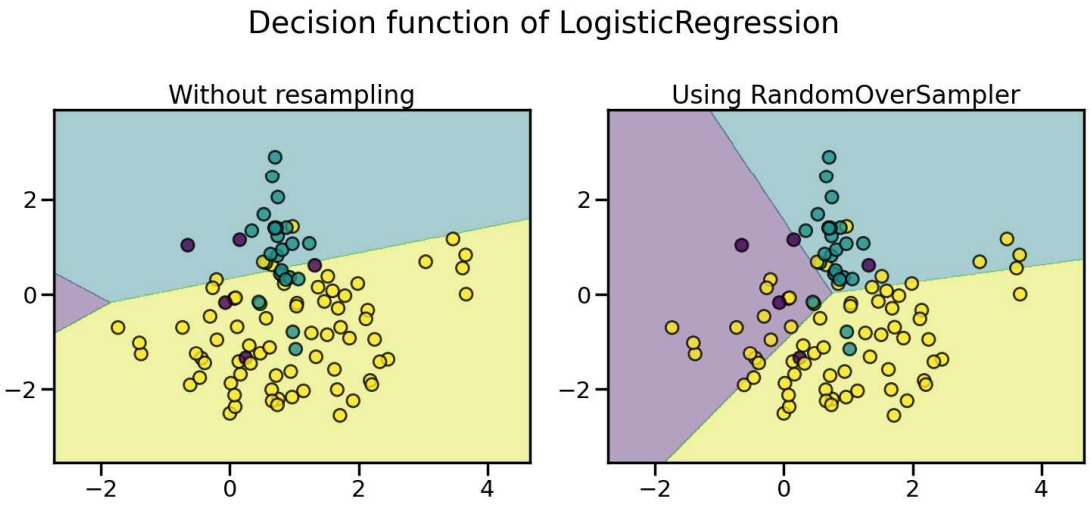

Figura 11. Comparativa entre la clasificación de una regresión logística sin y con Random Oversampling [19].

Como se puede observar, al aplicar el algoritmo de *Random Oversampling*, únicamente se duplican las instancias ya existentes de las clases minoritarias. Aunque se puede añadir cierta variabilidad con el parámetro *shrinkage* (Figura 12), esto podría, en ciertas situaciones, crear instancias falsas que, habitualmente, serian pertenecientes a la clase mayoritaria y son forzadas a ser minoritarias. Esto aumentaría el rendimiento sobre el papel, pero el modelo no sería representativo a la realidad [20].

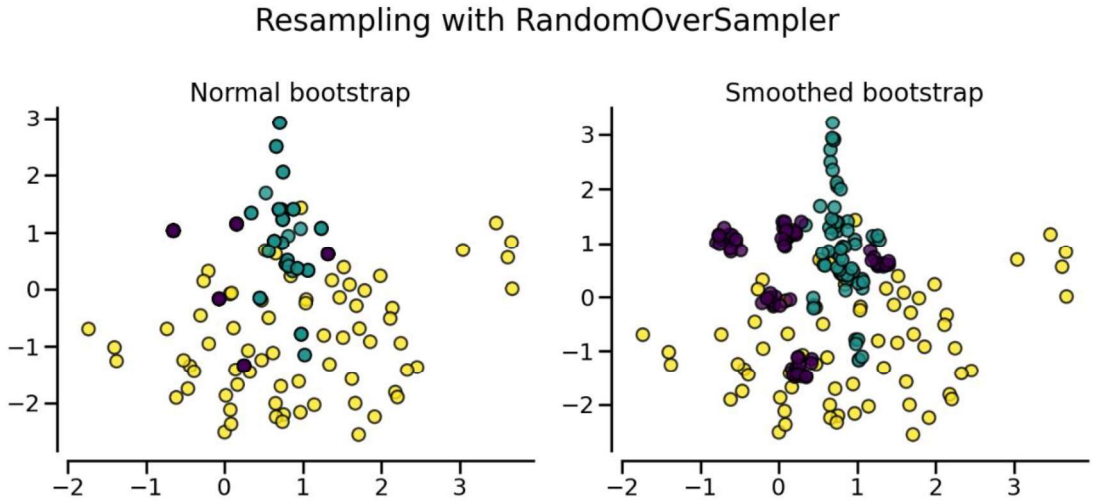

Figura 12. Efecto al aplicar variabilidad en los valores creados con Random Oversampling [19].

Para evitar resultados erróneos al aplicar *oversampling*, es necesario tener en cuenta ciertas consideraciones.

- En primer lugar, es muy importante solo aplicar el *oversampling* en el conjunto de datos de entrenamiento y mantener intacto el conjunto de datos de test. En el caso de que se aplicase el *oversampling* antes de dividir el conjunto de datos en entrenamiento y test, parte de las instancias duplicadas podrían acabar en ambos conjunto de datos, provocando que el modelo memorizase puntos específicos, sesgando el modelo y obteniendo resultados muy optimistas que no son representativos de la realidad [21].

- Por otra parte, aplicar un *oversampling* para balancear un conjunto de datos muy desequilibrado, en ciertas proporciones, puede generar que el modelo se ajuste demasiado a la clase minoritaria, lo que conlleva a un mayor número de falsos positivos [18].

- Por último, al generar nuevas instancias de la clase minoritaria de manera artificial, podemos obtener *overfitting* en el modelo. Se entiende como *overfitting* cuando el modelo únicamente entiende como válidos los datos con los que se ha entrenado. Esto provoca que, al intentar predecir una nueva instancia, este la categorice de manera errónea porque no es exactamente igual que los valores con los que se ha entrenado. Este fenómeno es muy habitual en el *Random Oversampling*, ya que, al duplicar varias veces ciertas instancias de la clase minoritaria, durante el entrenamiento, el modelo ve los mismos ejemplos una y otra vez, dándoles más importancia [22].

5. Resultados

Para realizar todas las pruebas y experimentos, se han seguido una serie de pasos. En primer lugar, se ha realizado el procesado de los datos tal y como se ha explicado en el apartado 3. Seguidamente, una vez obtenido el conjunto de datos con los datos procesados, se ha separado el conjunto de datos en entrenamiento y test, concretamente, 70% entrenamiento y 30% test. Por último, para la incorporación del algoritmo de regresión logística, se han definido un número máximo de iteraciones de 1.000 para evitar que el modelo no converja.

El conjunto de datos utilizado en este proyecto presenta diferentes tipos de variables. Una vez realizado el procesado de estas, podemos dividirlas en dos grupos, las variables textuales y no textuales. Concretamente, las variables textuales pueden variar mucho en número dependiendo del tamaño máximo de vocabulario utilizado, por lo que encontrar una combinación óptima entre los diferentes tipos de variables y seleccionar el número óptimo de ellas, es necesario para realizar unas buenas predicciones. Por ese motivo, durante este apartado, se plantea estudiar los resultados obtenidos para diferentes combinaciones de variables, parámetros y tamaños máximos de vocabulario.

Analizando las consecuencias que tendrían los resultados en una aplicación del mundo real, un anuncio fraudulento puede desencadenar en una estafa perjudicando mucho a la persona afectada. Es por ese motivo, que se ha decidido centrarse en la tasa de verdaderos positivos, es decir, el *Recall* para anuncios fraudulentos, que nos da información sobre la cantidad de falsos negativos obtenidos por el modelo. Aun así, obtener valores elevados de *Recall* sacrificando por completo la *Precision* tampoco es buena idea, ya que gran número de anuncios lícitos serán categorizados como fraudulentos.

5.1. Resultados con conjunto de datos sin *Oversampling*

Para comprobar si aplicar *oversampling* al conjunto de datos de entrenamiento es necesario, se realizarán diferentes pruebas con el conjunto de datos original. Además, para comprobar la importancia de las diferentes variables de este, se realizarán pruebas con uno formado únicamente por variables no textuales, otro con únicamente variables

textuales y, por último, uno con ambos tipos de variables. En este caso, en las situaciones donde se utilicen variables textuales, se fijará un vocabulario máximo de 500 palabras por variable textual. Aunque el valor óptimo puede ubicarse con un tamaño de vocabulario diferente, utilizar un número elevado de palabras aumenta las posibilidades de que el modelo encuentre patrones en el texto y, por lo tanto, nos muestre las diferencias entre los distintos conjuntos de datos.

La primera prueba realizada corresponde al conjunto de datos con variables no textuales, donde encontramos 190 variables. En la Tabla 6 se muestra la matriz de confusión obtenida y en la Tabla 7 las métricas obtenidas.

		Predicciones	
		Fraudulento (1)	**No Fraudulento (0)**
Valores Reales	**Fraudulento (1)**	51	140
	No Fraudulento (0)	15	2.972

Tabla 6. Matriz de confusión con un conjunto de datos de variables no textuales

	Fraudulento (1)
Precision	0,77
Recall	0,27
F1	0,40

Tabla 7. Métricas con un conjunto de datos de variables no textuales

Como se puede observar en la matriz de confusión, debido a que el conjunto de datos se encuentra desequilibrado, el algoritmo tiende a categorizar hacia la clase mayoritaria, es decir, los anuncios no fraudulentos. Por ese motivo, tanto la *Precision* como el *Recall* de las variables negativas (No Fraudulentos) presentan unos valores tan elevados. Por otra parte, con respecto a los valores positivos (Fraudulentos) el valor del *Recall* es muy bajo, debido a la tendencia a la categorización de valores negativos. El valor de *Precision*, en cambio, al presentar pocos falsos positivos, tiene un valor más alto.

La segunda prueba corresponde al conjunto de datos con únicamente variables textuales. En este caso el conjunto de datos presenta 2.286 variables. En la Tabla 8 y 9 se muestran la matriz de confusión y las métricas obtenidas, respectivamente.

		Predicciones	
		Fraudulento (1)	**No Fraudulento (0)**
Valores Reales	**Fraudulento (1)**	119	72
	No Fraudulento (0)	4	2.983

Tabla 8. Matriz de confusión con un conjunto de datos de variables textuales (500 palabras de vocabulario máximo por variable)

	Fraudulento (1)
Precision	0,97
Recall	0,62
F1	0,76

Tabla 9. Métricas con un conjunto de datos de variables textuales (500 palabras de vocabulario máximo por variable)

Como se puede observar en los resultados, el modelo sigue con la tendencia a categorizar respecto a los valores negativos. Respecto a los valores positivos, se ha obtenido una mejora respecto al anterior conjunto de datos y, por lo tanto, un valor superior de *Recall* y F1, 0,62 y 0,76, respectivamente.

Por último, la última prueba realizada ha sido con un conjunto de datos con variables textuales y no textuales, con un número de variables igual a 2.475. De la misma manera, en la Tabla 10 se muestra la matriz de confusión y en la Tabla 11 los resultados de las métricas obtenidos.

		Predicciones	
		Fraudulento (1)	**No Fraudulento (0)**
Valores Reales	**Fraudulento (1)**	141	52
	No Fraudulento (0)	11	2.976

Tabla 10. Matriz de confusión con un conjunto de datos de variables textuales y no textuales (500 palabras de vocabulario máximo por variable)

	Fraudulento (1)
Precision	0,93
Recall	0,74
F1	0,82

Tabla 11. Métricas con un conjunto de datos de variables textuales y no textuales (500 palabras de vocabulario máximo por variable)

Observando los resultados, se observa un mayor número de falsos fraudulentos respecto al anterior conjunto de datos, pero una disminución de falsos lícitos considerable. De la misma manera que en el resto de los conjuntos de datos, el modelo sigue teniendo una tendencia a categorizar la variable mayoritaria, provocando que el *Recall* de los valores positivos no sea demasiado elevado, del 0,74. El valor de *Precision* ha disminuido de 0,97 a 0,93, pero se compensa con el aumento tanto en *Recall* como F1.

Analizando los resultados obtenidos con los tres conjuntos de datos, se puede llegar a la conclusión que las variables textuales tienen más peso que las no textuales, pero en conjunto se obtienen mejores resultados. Por otra parte, se observa una tendencia a categorizar respecto a la clase mayoritaria debido al gran desequilibrio que el conjunto de datos presenta.

5.2. Resultados con conjunto de datos con *Oversampling*

En las pruebas sin *oversampling*, se ha observado que con un conjunto de datos con variables tanto textuales como no textuales se obtienen mejores resultados. Para determinar si los resultados mejoran realizando *oversampling*, se realizará una prueba inicial con un conjunto de datos de variables textuales y no textuales con un *oversampling* del 50/50. En la Tabla 12 se muestra la matriz de confusión, donde se puede observar que el número de falsos positivos ha aumentado considerablemente, pero en contraposición, el número de falsos negativos ha disminuido.

		Predicciones	
		Fraudulento (1)	**No Fraudulento (0)**
Valores Reales	**Fraudulento (1)**	169	22
	No Fraudulento (0)	68	2.919

Tabla 12. Matriz de confusión con un conjunto de datos de variables textuales y no textuales (500 palabras de vocabulario máximo por variable)

A partir de las métricas mostradas en la Tabla 13, se puede observar un aumento considerable en el *Recall* de valores fraudulentos y un descenso en la *Precision*. El F1 ha disminuido ligeramente debido al gran descenso de la precisión, que no ha podido compensarse completamente con el *Recall*. Aun así, el objetivo principal es obtener el mínimo número de falsos negativos, es decir, anuncios fraudulentos que se han categorizado como lícitos y que no serán eliminados de la plataforma de empleo.

	Fraudulento (1)
Precision	0,71
Recall	0,88
F1	0,79

Tabla 13. Métricas con un conjunto de datos de variables textuales y no textuales con un oversampling del 50/50 (500 palabras de vocabulario por variable)

Es importante entender que estos resultados pueden optimizarse con una mejor selección de variables, tamaños de vocabulario, parámetros del modelo o una proporción diferente en el *oversampling*.

Durante el apartado 4.4, se comentaron las consecuencias de aplicar un *oversampling* muy pronunciado en conjuntos de datos muy desequilibrados. En este caso, al aplicar un *oversampling* del 50/50, se han generado una gran cantidad de instancias en la clase minoritaria. Esto provoca que el modelo tienda a categorizar la clase negativa y, por lo tanto, que aparezcan más falsos negativos. Por ese motivo, aplicar *oversampling* con unas proporciones donde se generen menos instancias en la clase minoritaria puede resultar en un modelo con una mejor clasificación.

Por otra parte, se ha observado la importancia de las variables textuales, donde las anteriores pruebas han sido con un tamaño máximo de vocabulario aleatorio y fijo. Aunque se podría pensar que a mayor tamaño máximo de vocabulario más información se obtiene del texto, esto puede llegar a no ser cierto. En ciertas situaciones, tener demasiado vocabulario puede comportar un mayor número de variables y, por lo tanto, que al algoritmo le sea más difícil encontrar patrones para categorizar eficientemente.

Por estos motivos se ha decidido realizar una serie de pruebas donde se analizarán los resultados obtenidos con tamaños máximos de vocabulario y proporciones de *oversampling* diferentes. Concretamente, se estudiarán conjuntos de datos con un vocabulario máximo de 10 a 800 palabras y unas proporciones de 50/50, 66/33 y 75/25.

Antes de mostrar los resultados, es necesario definir qué situaciones queremos optimizar. Como se ha comentado anteriormente, se ha decidido optar por minimizar el número de falsos lícitos, por lo que se busca un *Recall* de los valores fraudulentos elevados. Esto supone un descenso de la precisión, por lo que se ha planteado otra opción donde se busca un *Recall* elevado, pero sin comprometer demasiado la *Precision*. Para ello, se buscará un valor de F1 elevado.

5.2.1. *Oversampling* con proporción del 50-50

Una vez planteadas las dos situaciones a optimizar, se han obtenido los resultados para diferentes tamaños máximos de vocabulario, de 10 a 800 palabras con un intervalo de 10 palabras. En la Figura 13 se muestra la evolución para el *Recall*, la *Precision* y el F1 de los valores fraudulentos. En la Tabla 14 se muestran los valores máximos para cada una de las métricas.

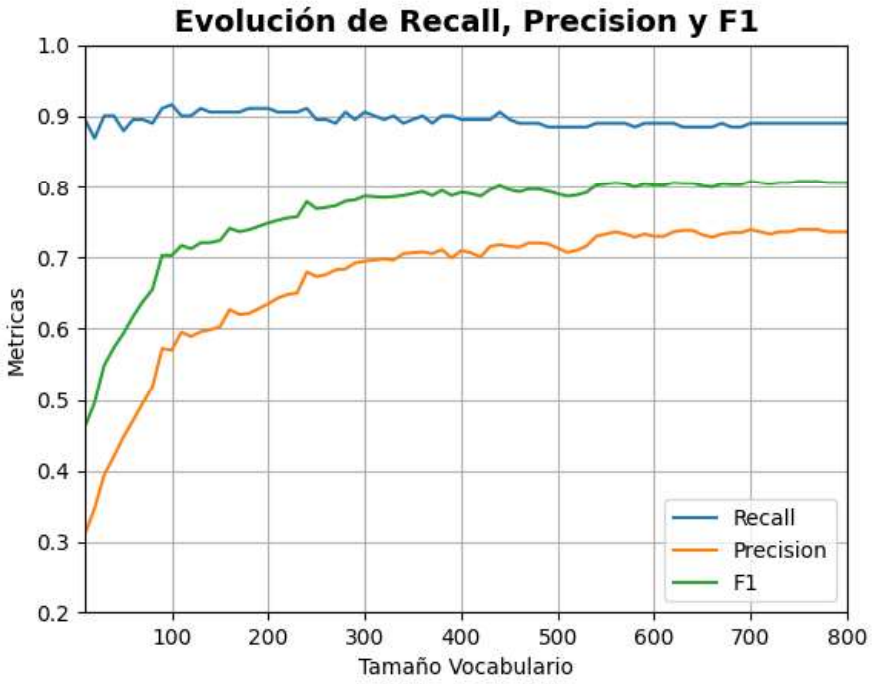

Figura 13. Evolución de Recall, Precision y F1 para vocabularios de 10 a 800 palabras y oversampling del 50-50

	Valor	**Tamaño de vocabulario**
Recall	0,92	100 palabras
Precision	0,74	700, 750, 760 y 770 palabras
F1	0,81	700, 750, 760 y 770 palabras

Tabla 14. Métricas máximas oversampling 50-50

Como se puede observar, se encuentra un punto máximo en el *Recall* sobre las 100 palabras de vocabulario con un valor de 0,92. Si se observa la matriz de confusión de la

Tabla 15, el valor de falsos positivos es elevado, provocando un valor de *Precision* bajo, tal y como se muestra en las métricas de la Tabla 16.

		Predicciones	
		Fraudulento (1)	**No Fraudulento (0)**
Valores Reales	**Fraudulento (1)**	175	16
	No Fraudulento (0)	132	2.855

Tabla 15. Matriz de confusión para un Recall máximo (100 palabras de vocabulario máximo por variable)

	Fraudulento (1)
Precision	0,57
Recall	0,92
F1	0,70

Tabla 16. Métricas con un conjunto de datos de variables textuales y no textuales con un oversampling del 50/50 (100 palabras de vocabulario máximo por variable)

Por otra parte, si buscamos un valor de F1 máximo, este se obtiene en las 700, 750, 760 y 770 palabras máximas de vocabulario. Para todos estos tamaños de vocabulario, los valores de las tres métricas son idénticos, por lo que se ha escogido el vocabulario máximo de 700 palabras. Como se puede observar en su matriz de confusión (Tabla 17), el número de falsos fraudulento ha disminuido considerablemente, obteniendo un valor de *Precision* de 0,74, 0,17 más elevado que en utilizar 100 palabras (Tabla 18). Aunque el valor de *Recall* haya disminuido ligeramente, esto comporta un aumento de únicamente 5 falsos lícitos.

		Predicciones	
		Fraudulento (1)	**No Fraudulento (0)**
Valores Reales	**Fraudulento (1)**	170	21
	No Fraudulento (0)	60	2.927

Tabla 17. Matriz de confusión para un F1 máximo (700 palabras de vocabulario máximo por variable)

	Fraudulento (1)
Precision	0,74
Recall	0,89
F1	0,81

Tabla 18. Métricas con un conjunto de datos de variables textuales y no textuales con un oversampling del 50/50 (700 palabras de vocabulario máximo por variable)

Con tal de reducir el tamaño máximo de vocabulario, se ha analizado la evolución de las métricas (Figura 12) con tal de encontrar puntos de interés con los que se obtengan predicciones similares. Para esta proporción de equilibrado, se observa que el punto con un tamaño de vocabulario máximo de 560 palabras tiene unos resultados similares al punto con F1 máximo. Según su matriz de confusión (Tabla 19), se observa que los falsos lícitos se mantienen y únicamente se obtiene un falso fraudulento más. Las métricas obtenidas, al presentar predicciones muy similares, se han mantenido prácticamente iguales (Tabla 20). En este caso, al mantener la misma cantidad de falsos negativos y únicamente aumentar en 1 el número de falsos positivos, se ha decidido decantarse por el punto con 560 palabras máximas de vocabulario.

		Predicciones	
		Fraudulento (1)	**No Fraudulento (0)**
Valores Reales	**Fraudulento (1)**	170	21
	No Fraudulento (0)	61	2.926

Tabla 19. Matriz de confusión para un F1 máximo (560 palabras de vocabulario máximo por variable)

	Fraudulento (1)
Precision	0,74
Recall	0,89
F1	0,81

Tabla 20. Métricas con un conjunto de datos de variables textuales y no textuales con un oversampling del 50/50 (560 palabras de vocabulario máximo por variable)

Con los datos obtenidos, en el caso de que el objetivo sea obtener el *Recall* máximo y el número de falsos fraudulentos no importe, se deberá utilizar un tamaño de vocabulario máximo de 100 palabras. Aun así, se recomienda utilizar el modelo con 560 palabras máximas de vocabulario, ya que la disminución del *Recall* es mínima, únicamente de 0,03, y el aumento de la *Precision* es considerable respecto a la obtenida con un vocabulario de 100 palabras.

5.2.2. *Oversampling* con proporción del 67-33

Una de las posibles razones por la que los resultados obtenidos no mejoran demasiado es la proporción de *oversampling* utilizada. El conjunto de datos original presenta una proporción aproximada de 93/7, por lo que aplicar una proporción 50/50 comporta un aumento muy elevado de las instancias de la clase minoritaria, que podría provocar un aumento de los falsos fraudulentos. Para confirmar si la hipótesis es correcta, se ha decidido realizar las mismas pruebas con una proporción de 67/33.

Observando la evolución de las métricas mostrada en la Figura 14, se observa que el *Recall* sigue manteniendo una evolución muy constante, pero con valores ligeramente inferiores a los obtenidos con la anterior proporción de *oversampling*. En la Tabla 21 se muestran los valores máximos para cada una de las métricas. El *Recall* presenta un valor máximo de 0.90 con un tamaño de vocabulario de 180 y 230 palabras máximas. Para la *Precsion* y F1, encontramos que sus valores máximos se encuentran en las 790 y 800 palabras máximas, con valores de 0,8 y 0,84, respectivamente.

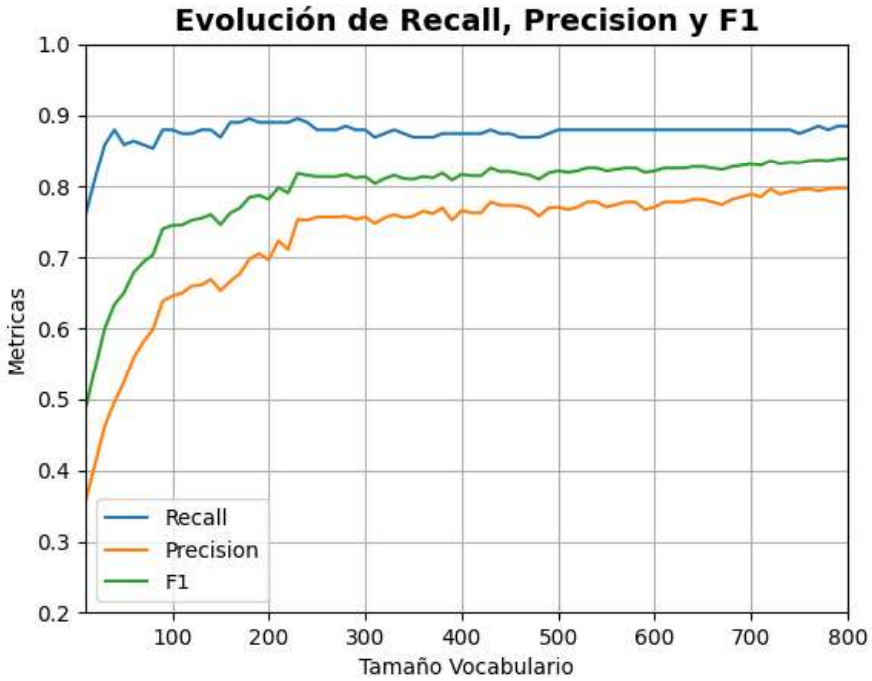

Figura 14. Evolución de Recall, Precision y F1 para vocabularios de 10 a 800 palabras y oversampling del 67-33

	Valor	**Tamaño de vocabulario**
Recall	0,90	180 y 230 palabras
Precision	0,80	790 y 800 palabras
F1	0,84	790 y 800 palabras

Tabla 21. Métricas máximas oversampling 67-33

Para *Recall* máximos, aunque este se encuentra en un tamaño de vocabulario máximo de 180 y 230 palabras, con 230 palabras el resto de las métricas son mayores, por lo que se ha analizado el modelo con ese tamaño máximo de vocabulario. Observando la matriz de confusión (Tabla 22), observamos una pequeña mejora en el número de falsos positivos respecto al punto de máximo F1 para *oversampling* de 50/50. Además, se observa un falso negativo menos, obteniendo en general unos mejores resultados. En la Tabla 23, se muestran los valores de las métricas para un *Recall* máximo, donde encontramos una mejora respecto a los valores obtenidos anteriormente.

		Predicciones	
		Fraudulento (1)	**No Fraudulento (0)**
Valores Reales	**Fraudulento (1)**	171	20
	No Fraudulento (0)	56	2.931

Tabla 22. Matriz de confusión para un Recall máximo (230 palabras de vocabulario máximo por variable)

	Fraudulento (1)
Precision	0,75
Recall	0,90
F1	0,82

Tabla 23. Métricas con un conjunto de datos de variables textuales y no textuales con un oversampling del 67/33 (230 palabras de vocabulario máximo por variable)

Si analizamos el punto donde F1 es máximo, observamos que este se presenta con dos tamaños máximos de vocabulario, 790 y 800 palabras. Como las métricas presentan valores iguales en ambos vocabularios, se ha escogido un vocabulario de 790 palabras máximas. Según los valores obtenidos en la matriz de confusión (Tabla 24), se puede observar una disminución de 13 falsos fraudulentos y un aumento de 2 falsos lícitos, respecto al modelo con 230 palabras. En la Tabla 25 se pueden observar los valores obtenidos en las métricas, donde se obtiene un aumento en la *Precision* y F1 respecto a la situación anterior de 0,05 y 0,02, respectivamente. El *Recall*, en cambio, al aumentar el número de falsos negativos, ha disminuido un 0,02 respecto a su valor máximo.

		Predicciones	
		Fraudulento (1)	**No Fraudulento (0)**
Valores Reales	**Fraudulento (1)**	169	22
	No Fraudulento (0)	43	2.944

Tabla 24. Matriz de confusión para un F1 máximo (790 palabras de vocabulario máximo por variable)

	Fraudulento (1)
Precision	0,80
Recall	0,88
F1	0,84

Tabla 25. Métricas con un conjunto de datos de variables textuales y no textuales con un oversampling del 67/33 (790 palabras de vocabulario máximo por variable)

Analizando la evolución de las métricas (Figura 13), se observan algunos puntos potenciales de obtener resultados similares en las predicciones, pero con un tamaño máximo de vocabulario menor. El primer punto corresponde a un vocabulario máximo de 500 palabras, que al obtener un aumento de un falso negativo y 7 falsos positivos, queda descartado. El segundo punto se encuentra en las 720 palabras máximas, donde se ha obtenido un falso negativo más que con 790 palabras, pero manteniendo los falsos positivos. Aunque hay mejora respecto al primer punto, los resultados obtenidos respecto al punto con F1 máximo son peores, con una disminución del vocabulario de únicamente 70 palabras, por lo que también se descarta este punto.

Para acabar, comparando los resultados obtenidos con respecto al *oversampling* del 50/50, encontramos una mejora clara en el número de falsos fraudulentos. Esto puede ser debido a que el modelo, al presentar un número menor de instancias repetidas en la clase minoritaria, tiene una tendencia menor a categorizarla. Por otra parte, comparando los valores obtenidos para ambas situaciones, se observa una diferencia menos significativa. Aun así, la mejora obtenida en la *Precision* para el F1 máximo compensa la disminución del *Recall*, por lo que el modelo obtenido con un tamaño máximo de vocabulario de 790 palabras, en general, obtiene mejores resultados.

5.2.3. *Oversampling* con proporción del 75-25

Se ha observado que con una proporción menor de la clase minoritaria, el modelo tiende a obtener mejores resultados. Aunque con los resultados obtenidos con una proporción de 67/33 se ha obtenido una mejora, en esta última prueba se platea aplicar una proporción de *oversampling* aún menor respecto a la clase minoritaria y analizar cómo afecta al modelo. De la misma manera que en el resto de los conjuntos de datos, se aplicaran las mismas pruebas con la finalidad de poder comparar los resultados.

En la Figura 15 se muestra la evolución del *Recall*, *Precision* y F1. Como se puede observar, los valores obtenidos en el *Recall* son inferiores al resto de pruebas, obteniendo su valor máximo de 0,88 con un tamaño de vocabulario de 190 y 210. Aun así, se observa un aumento en la *Precision* en todo el rango de vocabularios máximos. El valor máximo de F1 lo encontramos aproximadamente entorno a las 770 y 780 palabras máximas con un valor de 0,85 (Tabla 26).

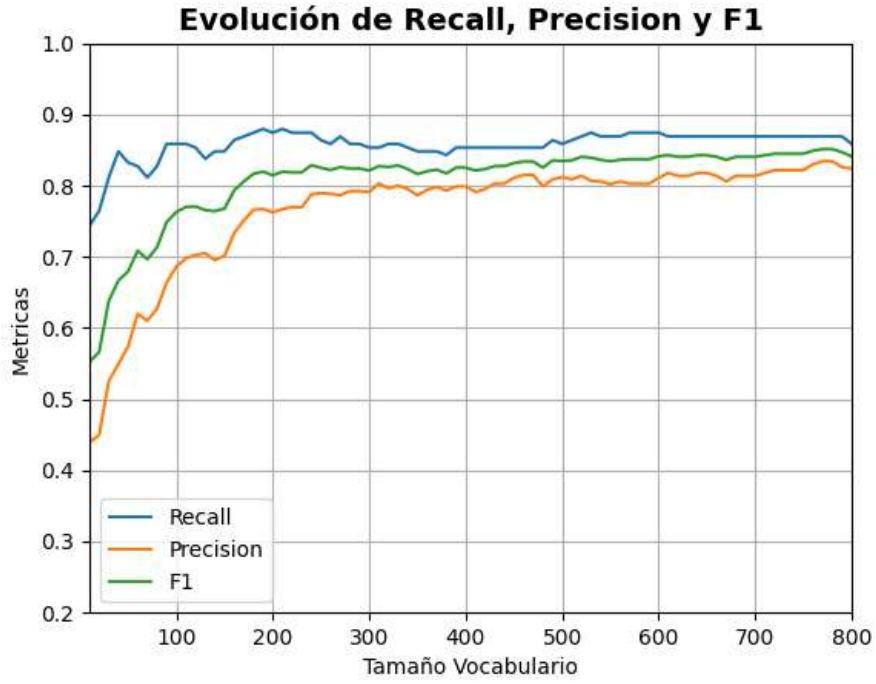

Figura 15. Evolución de Recall, Precision y F1 para vocabularios máximos de 10 a 800 palabras y oversampling del 75-25

	Valor	Tamaño de vocabulario
Recall	0,88	190 y 210 palabras
Precision	0,83	770 y 780 palabras
F1	0,85	770 y 780 palabras

Tabla 26. Métricas máximas oversampling 75-25

En la primera situación, donde se busca un *Recall* máximo, este presenta un valor de 0,88 y se obtiene en las 190 o 210 palabras máximas. Debido a que el valor del resto de métricas es idéntico para los dos tamaños de vocabulario, de la misma manera que en situaciones anteriores, se ha escogido el modelo con menos palabras, 190. A partir de aquí, analizando su matriz de confusión (Tabla 27) y los resultados del resto de métricas para ese tamaño de vocabulario (Tabla 28), se observa que, respecto a los resultados obtenidos con F1 máximo para una proporción de 67-33, se presenta un ligero empeoramiento en los resultados. Se han obtenido 8 falsos fraudulentos y 1 falso líticito más.

		Predicciones	
		Fraudulento (1)	**No Fraudulento (0)**
Valores Reales	**Fraudulento (1)**	168	23
	No Fraudulento (0)	51	2.936

Tabla 27. Matriz de confusión para un Recall máximo (190 palabras de vocabulario máximo por variable)

	Fraudulento (1)
Precision	0,77
Recall	0,88
F1	0,82

Tabla 28. Métricas con un conjunto de datos de variables textuales y no textuales con un oversampling del 75/25 (190 palabras de vocabulario máximo por variable)

Si buscamos maximizar el valor de F1, este se encuentra con un tamaño máximo de vocabulario de 770 o 780 palabras y un valor de 0,85. Si analizamos la matriz de confusión con un tamaño máximo de vocabulario de 770 palabras (Tabla 29), se observa un aumento de 3 falsos lícitos respecto a los resultados obtenidos con una proporción de 67-33. Por otra parte, el número de falsos fraudulentos ha disminuido en 10. Esto comporta una disminución del *Recall* de 0,01 y un aumento en la *Precision* y el F1 de 0,07 y 0,032, respectivamente (Tabla 30).

		Predicciones	
		Fraudulento (1)	**No Fraudulento (0)**
Valores Reales	**Fraudulento (1)**	166	25
	No Fraudulento (0)	33	2.954

Tabla 29. Matriz de confusión para un F1 máximo (770 palabras de vocabulario máximo por variable)

	Fraudulento (1)
Precision	0,83
Recall	0,87
F1	0,85

Tabla 30. Métricas con un conjunto de datos de variables textuales y no textuales con un oversampling del 75/25 (770 palabras de vocabulario máximo por variable)

Analizando los resultados obtenidos con esta proporción de equilibrado, se llega a la conclusión de que el modelo que busca un F1 máximo, presenta un mejor equilibrio entre los falsos positivos y negativos obtenidos y presenta unas mejores predicciones.

Para acabar, de la misma manera que en las pruebas anteriores, a partir de la Figura 14, se ha observado que el punto con un tamaño máximo de vocabulario de 600 palabras por variable obtiene un resultado similar al de 770 palabras. Analizando su matriz de confusión (Tabla 31) y sus métricas (Tabla 32), se observa que el modelo ha predicho 1 falso lícito menos, sacrificando predecir 6 falsos fraudulentos más. En cuanto a las métricas, el *Recall* toma practicante el mismo valor y la *Precision* y el F1 han disminuido 0,02 y 0,01, respectivamente.

		Predicciones	
		Fraudulento (1)	**No Fraudulento (0)**
Valores Reales	**Fraudulento (1)**	167	24
	No Fraudulento (0)	39	2.948

Tabla 31. Matriz de confusión para un F1 máximo (600 palabras de vocabulario máximo por variable)

	Fraudulento (1)
Precision	0,81
Recall	0,87
F1	0,84

Tabla 32. Métricas con un conjunto de datos de variables textuales y no textuales con un oversampling del 75/25 (600 palabras de vocabulario máximo por variable)

Otro punto de interés es el modelo con un tamaño máximo de vocabulario de 530 palabras. En este caso, se han obtenido peores resultados, ya que se ha predicho 1 falso negativo más, pero sin reducir la cantidad de falsos positivos. Comparando resultados entre los puntos de 600 y 770 palabras de vocabulario, se llega a la conclusión que el aumento de falsos negativos no compensa por la disminución de un único falso positivo, por lo que el modelo con un tamaño máximo de vocabulario de 770 palabras, en general, obtiene mejores predicciones.

Comparando los valores obtenidos para cada una de las pruebas, se llegan a una serie de conclusiones. En primer lugar, las situaciones donde se busca un *Recall* máximo sin importar los valores de las otras métricas, en general, obtienen peores resultados que cuando se busca un equilibrio, es decir, se maximiza F1. En segundo lugar, se ha observado mejores resultados a medida que las proporciones de *oversampling* favorecían a la clase mayoritaria, ya que el número de falsos lícitos disminuía considerablemente sin comprometer demasiado a los falsos fraudulentos.

Con estas consideraciones en cuenta y siguiendo el objetivo marcado al inicio del apartado, se ha escogido el conjunto de datos con variables textuales y no textuales, con

proporción 75-25 y maximizando el valor de F1, es decir, un máximo de 770 palabras de vocabulario por variable.

5.2.4. Análisis del conjunto de datos final

Como ya se comentó en el apartado 4.1.3, a partir de la probabilidad calculada por el modelo, si esta se encuentra por encima de un valor de confianza, el anuncio se clasificará como fraudulento, mientras que, si esta es menor se clasificara como licito. Para encontrar el equilibrio perfecto en el conjunto de datos seleccionado, se analizará la evolución de la *Precision* y el *Recall* para diferentes valores de confianza. En la Figura 16 se muestra la curva para un modelo con una proporción de *oversampling* de 75-25 y un máximo de 770 palabras de vocabulario por variable textual.

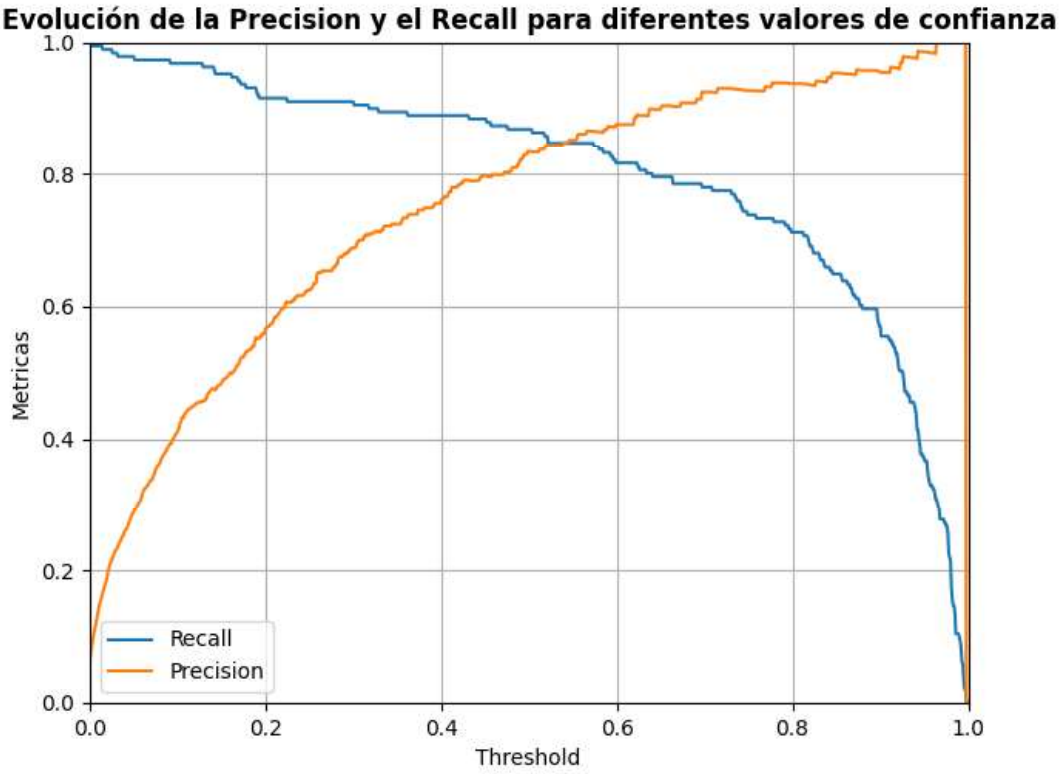

Figura 16. Evolución de la Precision y el Recall en todo el rango de valores de confianza

Como se puede observar, el punto donde la *Precision* y el *Recall* toman el mismo valor se encuentra para un valor de confianza de aproximadamente 0,54. Si se analizan su matriz de confusión y sus métricas (Tablas 33 y 34), se observa que, lógicamente, se presentan los mismos falsos positivos que negativos. Esto comporta que los falsos lícitos aumenten en 4 y los falsos fraudulentos disminuyan en 4. Con referencia a las métricas, el *Recall* disminuye en 0,02, mientras que la *Precision* aumenta en 0,02.

		Predicciones	
		Fraudulento (1)	**No Fraudulento (0)**
Valores Reales	**Fraudulento (1)**	162	29
	No Fraudulento (0)	29	2.958

Tabla 33. Matriz de confusión para un valor de confianza de 0,54

	Fraudulento (1)
Precision	0,85
Recall	0,85
F1	0,85

Tabla 34. Métricas obtenidas con un valor de confianza del 0,54

Aunque el anterior modelo encuentre un equilibrio perfecto entre falsos positivos y negativos, este punto no tiene por qué ser el idóneo para esta situación. En nuestro caso, se busca minimizar el número de falsos lícitos que el modelo predice, por lo que puede ser interesante estudiar puntos donde el *Recall* toma valores más elevados, sin comprometer demasiado a la *Precision*.

Con un valor de confianza entre 0,44 y 0,48, se observa claramente que el valor de la *Precision* se mantiene prácticamente constante. Aun así, a medida que el umbral disminuye, el *Recall* aumenta, por lo que sería interesante analizar el modelo con un valor de confianza de 0,44. Si analizamos su matriz de confusión (Tabla 35) y sus métricas (Tabla 36), se puede observar que únicamente se han obtenido una disminución de 3 falsos negativos, pero los falsos positivos han aumentado en 12 respecto a los valores de

la Tabla 29. En este caso, el *Recall* ha aumentado en 0,01, la *Precision* ha disminuido en 0,02 y el F1 ha disminuido en 0,02.

		Predicciones	
		Fraudulento (1)	**No Fraudulento (0)**
Valores Reales	**Fraudulento (1)**	169	22
	No Fraudulento (0)	45	2.942

Tabla 35. Matriz de confusión para un valor de confianza de 0,44

	Fraudulento (1)
Precision	0,79
Recall	0,88
F1	0,83

Tabla 36. Métricas obtenidas con un valor de confianza del 0,44

Por último, otro punto interesante se encuentra en un valor de confianza aproximado de 0,57. En este punto, según la Figura 16, el valor del *Recall* toma un valor muy similar al primer punto analizado, pero la *Precision* toma un valor más elevado. A partir de la matriz de confusión (Tabla 37) y las métricas obtenidas (Tabla 38), se ha obtenido un aumento de 4 falsos lícitos y una disminución de 8 falsos fraudulentos respecto a los valores de la Tabla 29. En consecuencia, el *Recall* ha disminuido en 0,02, la *Precision* ha aumentado en 0,04 y el F1 ha aumentado en 0,01.

		Predicciones	
		Fraudulento (1)	**No Fraudulento (0)**
Valores Reales	**Fraudulento (1)**	162	29
	No Fraudulento (0)	25	2.962

Tabla 37. Matriz de confusión para un valor de confianza de 0,57

	Fraudulento (1)
Precision	0,87
Recall	0,85
F1	0,86

Tabla 38. Métricas obtenidas con un valor de confianza del 0,57

Analizando los diferentes puntos obtenidos, se observa que el punto donde la *Precision* y el Recall son iguales obtiene muy buenos resultados, ya que, aunque aumenten los falsos negativos, esto se compensa con una disminución de los falsos positivos. Por otra parte, el punto con un valor de confianza de 0,44 queda descartado debido a que la disminución de los falsos negativos no compensa el aumento más pronunciado de falsos positivos. Por último, los mejores resultados se han obtenido con un valor de confianza de 0,57, ya que aunque los falsos negativos aumentasen, se ha producido una disminución mayor de falsos positivos. Por lo tanto, se utilizará el modelo con una proporción de *oversampling* de 75-25, 770 palabras de vocabulario por variable textual y un valor de confianza de 0,57.

Por último, para saber la importancia del vocabulario y otras variables en el conjunto de datos, se analizarán los coeficientes obtenidos por el modelo. Cabe destacar, que para estudiar la importancia de cada una de las variables, es necesario obtener los coeficientes en valor absoluto. Ordenando estos coeficientes, se observa que un gran número de variables tienen un coeficiente con un valor bajo. Para observar cómo afecta al modelo la eliminación de variables poco significativas, se plantean una serie de pruebas. La primera prueba consiste en eliminar todas las variables con coeficientes menores a 0,01 en valor absoluto. En la Tabla 39 y 40 se observan los resultados obtenidos para un modelo con 3.468 variables, es decir, 87 variables menos respecto a las 3.555 variables iniciales. Los resultados obtenidos son peores respecto al modelo de partida, ya que se han obtenido 2 falsos negativos y un falso positivo más, con únicamente una disminución de 87 variables.

		Predicciones	
		Fraudulento (1)	**No Fraudulento (0)**
Valores Reales	**Fraudulento (1)**	160	31
	No Fraudulento (0)	26	2.961

Tabla 39. Matriz de confusión para variables con coeficientes mayores de 0,01

	Fraudulento (1)
Precision	0,86
Recall	0,84
F1	0,85

Tabla 40. Métricas obtenidas para variables con coeficientes mayores de 0,01

Si probamos a eliminar las variables con coeficientes menores que 0,05, se obtienen 3.156 variables, es decir, 399 variables menos. Con este modelo, se obtienen los resultados mostrados en la Tabla 41 y 42. Como se puede observar, se observa un mayor número de falsos positivos y negativos respecto al modelo de partida, concretamente, 5 falsos fraudulentos y 6 falsos lícitos más. Las métricas de *Recall*, *Precision* y F1 han disminuido todas un 0,03. Por lo tanto, este punto también se descarta, ya que se obtienen peores resultados por una disminución muy pequeña del número de variables.

		Predicciones	
		Fraudulento (1)	**No Fraudulento (0)**
Valores Reales	**Fraudulento (1)**	156	35
	No Fraudulento (0)	30	2.957

Tabla 41. Matriz de confusión para variables con coeficientes mayores de 0,05

	Fraudulento (1)
Precision	0,84
Recall	0,82
F1	0,83

Tabla 42. Métricas obtenidas para variables con coeficientes mayores de 0,05

Para acabar, se hará una última prueba eliminando las variables con coeficientes inferiores a 0,1, para observar si se eliminan muchas variables sin perjudicar demasiado al rendimiento del modelo. En este caso, se han obtenido 2.726 variables, 829 menos. En la Tabla 43 y 44 se muestran los resultados obtenidos, donde observamos que el modelo ha predicho mejor que en la situación anterior. Aun así, presenta 6 falsos lícitos y 2 falsos fraudulentos más.

		Predicciones	
		Fraudulento (1)	**No Fraudulento (0)**
Valores Reales	**Fraudulento (1)**	156	35
	No Fraudulento (0)	27	2.960

Tabla 43. Matriz de confusión para variables con coeficientes mayores de 0,1

	Fraudulento (1)
Precision	0,85
Recall	0,82
F1	0,83

Tabla 44. Métricas obtenidas para variables con coeficientes mayores de 0,1

Con los resultados obtenidos, se llega a la conclusión que reducir el número de variables, en ciertos casos, aunque estas el modelo no las considere importantes debido a su coeficiente bajo, provoca unos resultados significativamente peores. Aun así, el modelo para variables con coeficientes mayores a 0,1 ha obtenido resultados parecidos con una disminución considerable en las variables. De todas maneras, el tiempo de cómputo para el modelo de partida no es muy elevado, por lo que se ha decidido escoger el modelo con una proporción de *oversampling* del 75-25, un valor de confianza del 0,57 y un tamaño máximo de vocabulario de 770 palabras, obteniendo un total de 3.555 variables.

5.3. Resumen de los resultados

Para entender mejor los resultados obtenidos durante las pruebas realizadas, se han agrupado los mejores resultados en la siguiente tabla (Tabla 45), En ella se muestran las características principales del conjunto de datos utilizado para ajustar el modelo, los valores de las métricas para anuncios fraudulentos y su matriz de confusión.

Características del conjunto de datos	**Métricas**	**Matriz de confusión**
Proporción de *oversampling* del 67-33, tamaño máximo de vocabulario de 790 palabras y valor de confianza de 0,5	*Precision* = 0,80 *Recall* = 0,88 F1 = 0,84	Verdaderos fraudulentos = 169 Verdaderos lícitos = 2.944 Falsos fraudulentos = 43 Falsos lícitos = 22
Proporción de *oversampling* del 75-25, tamaño máximo de vocabulario de 770 palabras y valor de confianza de 0,5	*Precision* = 0,83 *Recall* = 0,87 F1 = 0,85	Verdaderos fraudulentos = 166 Verdaderos lícitos = 2.954 Falsos fraudulentos = 33 Falsos lícitos = 25
Proporción de *oversampling* del 75-25, tamaño máximo de vocabulario de 770 palabras y valor de confianza de 0,54	*Precision* = 0,85 *Recall* = 0,85 F1 = 0,85	Verdaderos fraudulentos = 162 Verdaderos lícitos = 2.958 Falsos fraudulentos = 29 Falsos lícitos = 29

Proporción de *oversampling* del 75-25, tamaño máximo de vocabulario de 770 palabras y valor de confianza de 0,57	*Precision* = 0,87 *Recall* = 0,85 F1 = 0,86	Verdaderos fraudulentos = 162 Verdaderos lícitos = 2.962 Falsos fraudulentos = 25 Falsos lícitos = 29

Tabla 45. Resumen de los mejores resultados obtenidos

Como se puede observar, los mejores resultados obtenidos corresponden a proporciones de *oversampling* del 67-33 y 75-25 y tamaños máximos de vocabulario elevados donde se maximiza el F1. De todos estos resultados, el que obtiene unas predicciones más equilibradas entre falsos fraudulentos y falos lícitos es el modelo con una proporción de equilibrado del 75-25, un tamaño máximo de vocabulario de 770 palabras por variable de texto y un valor de confianza de 0,57.

6. Planificación

La planificación del proyecto se muestra en el diagrama de Gantt de la Figura 17. En la Tabla 46 se muestran las diferentes etapas del proyecto, indicando su fecha de inicio, final y duración.

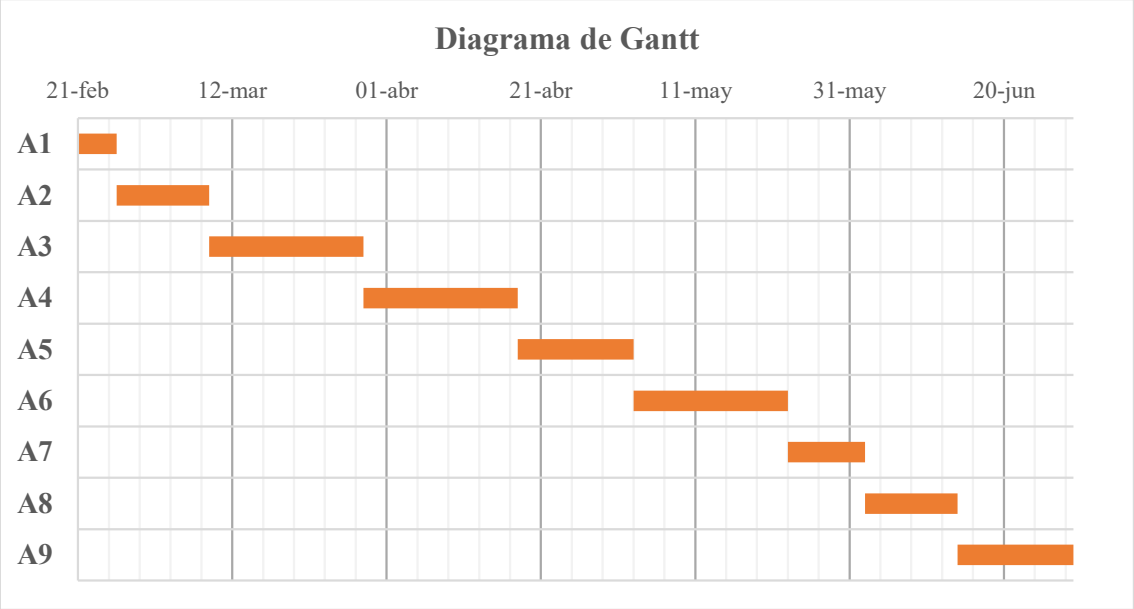

Figura 17. Diagrama de Gantt del proyecto

Actividades	Fecha de inicio	Duración (días)	Fecha final
A1. Introducción	21-feb	5	26-feb
A2. Comprensión de datos	26-feb	12	09-mar
A3. Procesado de datos	09-mar	20	29-mar
A4. Marco teórico	29-mar	20	18-abr
A5. Aplicación del modelo	18-abr	15	03-may
A6. Obtención y análisis de resultados	03-may	20	23-may

A7. Estudio económico y medioambiental	23-may	10	02-jun
A8. Conclusiones	02-jun	12	14-jun
A9. Revisión	14-jun	15	29-jun

Tabla 46. Actividades del proyecto con su fecha de inicio, duración y fecha final

7. Estudio económico

Para el estudio económico de este proyecto se deben tener en cuenta una serie de factores. En primer lugar, encontramos el personal experto en *data science* que ejecutará todo el proyecto. En este caso, se ha estimado una duración del proyecto de aproximadamente 100 horas. Estas horas se cobrarán según el salario medio de un ingeniero de *data science* en España, 35.431 € anuales [23], es decir, aproximadamente 16 €/hora. En segundo lugar, hay que tener en cuenta la amortización del material utilizado, el cual se supone una amortización del 20 % del coste del material. Por último, se tiene en cuenta la luz consumida durante todo el proyecto. Suponiendo un consumo del ordenador de 250 W y 40 W por las luces, multiplicado por las 100 horas del proyecto y por el precio medio de la luz (0,2807 €/kWh), se obtiene un importe de 8,14 €. En la Tabla 47 se muestra desarrollado el presupuesto del proyecto.

Nombre entregable	Cantidad	Coste Unitario	Coste Total
Ingeniero Data science	100 h	16 €/hora	1.600 €
Amortización material	100 h	-	200 €
Luz	29 kWh	0,2807 €/kWh	8,14 €
TOTAL			**1.808,14 €**

Tabla 47. Presupuesto desglosado del proyecto

8. Estudio medioambiental

Debido a que se trata de un proyecto digital, su impacto medioambiental se puede considerar prácticamente nulo. Aun así, se pueden tener en cuenta la electricidad consumida en el proceso de desarrollo del modelo y por otros dispositivos eléctricos. Suponiendo una potencia media de 290 W y una duración de 100 horas, esto supone un consumo medio de 29 kWh. Este consumo supone una cantidad ínfima de las emisiones que la producción de energía produce. Por ese motivo, podemos considerar que este proyecto no repercute a nivel medioambiental.

9. Conclusiones

Respecto a los objetivos planteados al inicio del proyecto, estos se han cumplido completamente. Durante el proyecto se han visto las fases de la metodología CRISP-DM que permiten realizar un proyecto de *data science* de manera eficiente. Con esto se ha conseguido obtener un modelo predictivo que permite detectar anuncios de trabajo fraudulentos a partir de un conjunto de datos formado por variables textuales y no textuales. Una de las primeras fases de la metodología corresponde a la compresión de datos, donde se han estudiado tanto individualmente como colectivamente cada una de las variables. Después, se ha realizado el procesado del conjunto de datos para que este pueda ser utilizado en el modelo de regresión logística. Con esto, se ha podido implementar el modelo y realizar una serie de pruebas.

Primeramente, al tener un conjunto de datos formado por variables textuales y no textuales y que, además, se encuentra desequilibrado, se han realizado pruebas para encontrar la mejor combinación de variables en la situación original. Aquí, se ha llegado a la conclusión de que un conjunto de datos formado por variables textuales y no textuales, en general, obtiene mejores resultados que el resto de las combinaciones.

En segundo lugar, para determinar si realizar un *oversampling* para equilibrar el conjunto de datos es una buena aproximación, se han realizado pruebas con un conjunto de datos con una proporción de 50/50 utilizando el algoritmo de *Random Oversampling*. Con estas pruebas, se ha visto que al aplicar *oversampling* los resultados obtenidos por el modelo eran mejores.

Una vez llegada a la conclusión de que aplicar *oversampling* para equilibrar el conjunto de datos daba lugar a un modelo con mejores predicciones, se realizaron pruebas para observar cómo se comportaba el modelo para proporciones de equilibrado diferentes. Concretamente, se han hecho pruebas con proporciones del 50/50, 67/33 y 75/25. Además, para cada una de las proporciones, se han analizado los resultados para diferentes tamaños máximos de vocabulario, de 10 a 800 palabras. Con estas pruebas, se ha observado que los puntos de *Recall* máximo, generalmente, obtienen peores resultados que en los puntos de F1 máximo. También se ha visto que para proporciones que favorecen la clase mayoritaria, 67/33 y 75/25, el modelo tiende a predecir mejor. De todas

estas opciones, se ha visto que el conjunto de datos que mejores resultados ha obtenido es uno con una proporción del 75/25 y maximizando el F1.

Por último, se ha hecho un análisis del conjunto de datos escogido durante las últimas pruebas. En este análisis, se ha estudiado el comportamiento del modelo para diferentes valores de confianza. Con el objetivo de obtener un *Recall* elevado, sin disminuir demasiado la *Precision*, se ha obtenido que con un valor de confianza del 0,57 el modelo predice algún falso lícito más, pero disminuyendo considerablemente el número de falsos fraudulentos. Para acabar, se ha estudiado la posibilidad de eliminar las variables con coeficientes prácticamente nulos, para observar si el modelo predecía igual o mejor, pero con un número de variables menor. Los resultados obtenidos no han sido buenos, ya que el rendimiento disminuye y la reducción de variables no es muy elevada. Con los resultados obtenidos, se llega a la conclusión que el conjunto de datos con una proporción del 75/25, con 3.555 variables y con un valor de confianza de 0,57, es el que presenta mejore resultados.

Aun así, el modelo no es perfecto, por lo que se podrían realizar mejoras en futuros proyectos. En primer lugar, un análisis más exhaustivo de selección de variables sería interesante, ya que, aunque parece que el gran número de variables no afecta al modelo utilizado, utilizar métodos de selección de variables más sofisticados podría comportar en un modelo más eficiente. Por otra parte, se podrían aplicar otros modelos más habituales en el *text mining* como el SVM (*Support Vector Machine*), *Random Forest*, *Boosting* o árboles de decisión. Por último, se podrían probar otros métodos para el procesado de lenguaje natural, como por ejemplo el *bag of words*.

Bibliografía

[1] What is Data Science?. (2022). https://www.oracle.com/data-science/what-is-data-science/

[2] What is Data Science & How Does Data Science Works?. (2022). https://www.mygreatlearning.com/blog/what-is-data-science/

[3] What is Machine Learning?. (2022). IBM Cloud Education. https://www.ibm.com/cloud/learn/machine-learning

[4] CRISP-DM: La metodología para poner orden en los proyectos - Sngular. (2022). https://www.sngular.com/es/data-science-crisp-dm-metodologia/

[5] CRISP-DM - Data Science Process Alliance. (2022). https://www.datascience-pm.com/crisp-dm-2/

[6] Applicant Tracking Systems: A Guide for Job Seekers. (2022). https://www.jobscan.co/applicant-tracking-systems

[7] Vidros, S., Kolias, C., Kambourakis, G., & Akoglu, L. (2017). Automatic Detection of Online Recruitment Frauds: Characteristics, Methods, and a Public Dataset. Future Internet.

[8] Chi-Square Test of Independence. (2022). https://www.jmp.com/en_au/statistics-knowledge-portal/chi-square-test/chi-square-test-of-independence.html

[9] 5.2 V de Cramer. (2022). https://www.uv.es/webgid/Descriptiva/52_v_de_cramer.html

[10] IBM Docs. (2022). https://www.ibm.com/docs/es/cognos-analytics/11.1.0?topic=terms-cramrs-v

[11] What is Natural Language Processing? (2022). IBM Cloud Education. https://www.ibm.com/cloud/learn/natural-language-processing

[12] Ramadhan, L. (2021). TF-IDF Simplified. https://towardsdatascience.com/tf-idf-simplified-aba19d5f5530

[13] Brownlee, J. (2022). Logistic Regression for Machine Learning. https://machinelearningmastery.com/logistic-regression-for-machine-learning/

[14] Febrianti, R., Widyaningsih, Y., & Soemartojo, S. (2021). The parameter estimation of logistic regression with maximum likelihood method and score function modification. *Journal Of Physics: Conference Series*, *1725*(1), 012014. doi: 10.1088/1742-6596/1725/1/012014

[15] Brownlee, J. (2022). Train-Test Split for Evaluating Machine Learning Algorithms. https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/

[16] Arce, J. (2022). La matriz de confusión y sus métricas – Inteligencia Artificial. https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/

[17] Lahera, G. (2019). Unbalanced Datasets & What To Do. https://medium.com/strands-tech-corner/unbalanced-datasets-what-to-do-144e0552d9cd

[18] Brownlee, J. (2020). Random Oversampling and Undersampling for Imbalanced Classification. https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/

[19] 2. Over-sampling — Version 0.9.1. (2022). https://imbalanced-learn.org/stable/over_sampling.html#random-over-sampler

[20] Tarawneh, A., Hassanat, A., Altarawneh, G., & Almuhaimeed, A. (2022). Stop Oversampling for Class Imbalance Learning: A Review.

[21] Brownlee, J. (2022). How to Avoid Data Leakage When Performing Data Preparation. https://machinelearningmastery.com/data-preparation-without-data-leakage/

[22] Overfitting. Qué es, causas, consecuencias y cómo solucionarlo | Grupo Atico34. (2022). https://protecciondatos-lopd.com/empresas/overfitting/

[23] Glassdoor. (2022). https://www.glassdoor.es/Sueldos/data-scientist-sueldo-SRCH_KO0,14.htm