UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

MÁSTER EN INGENIERÍA DE TELECOMUNICACIÓN

SISTEMA PREDICTIVO DE PROBABILIDAD DE COMPRA EN E-COMMERCE

Autor: Raúl Barba Alonso

Tutor: Juan Carlos Aguado Manzano

TRABAJO FIN DE MÁSTER

TÍTULO:	**Sistema predictivo de probabilidad de compra en e-commerce**
AUTOR:	**D. Raúl Barba Alonso**
TUTOR:	**Dr. D. Juan Carlos Aguado Manzano**
DEPARTAMENTO:	**Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

TRIBUNAL

PRESIDENTE:	**Ignacio de Miguel Jiménez**
VOCAL:	**Ramón José Durán Barroso**
SECRETARIO:	**Ramón de la Rosa Steinz**
SUPLENTE:	**Patricia Fernández Reguero**
SUPLENTE:	**Javier Aguiar Pérez**
SUPLENTE:	**María Jesús Verdú Pérez**

Resumen

El comercio digital es la nueva forma de obtener productos en la actualidad debido a los avances tecnológicos e internet. En paralelo surgió el marketing digital lanzando campañas publicitarias en las páginas web para aumentar la propensión de compra. Sin embargo, esto ha hecho que los usuarios sufran una sobreestimulación y que no adquieran sus productos. Por ello, en este proyecto se ha propuesto un sistema en el que se realice una predicción personalizada y en tiempo real de la probabilidad de compra. En tiempo real referido a obtener la probabilidad de compra de un cliente y producto determinado tras la finalización de cada acción del usuario, como puede ser un click en la pantalla del ordenador. Así, se podrán personalizar las campañas publicitarias conforme a las necesidades de los usuarios. El proyecto pretende implementar el sistema de predicción en tiempo real mencionado y compararlo con el modelo de predicción utilizado anteriormente basado en las predicciones que se realizan tras finalizar la sesión de los usuarios, es decir, a posteriori. A partir del sistema desarrollado se analizan diferentes métricas como la precisión y el recall, además de diferentes curvas para ver el comportamiento que presenta, como la curva PR y ROC.

Palabras clave

Marketing digital, predicción, tiempo real, precisión, recall, curva ROC.

Abstract

Nowadays, due to technological development and the internet, digital commerce is the prominent way to obtain products. At the same time, the digital marketing industry emerged, and started launching advertising campaigns on web pages to increase propensity to buy. However, this has made the users suffer from overstimulation, and consequently they are not acquiring their products. Therefore, this project proposes a system that performs real time personalized purchase likelihood. Real time meaning to obtain the probability of a purchase and a specific product after the completion of each user action, for instance, a click in the computer screen. Thus, advertising campaigns can be customised to the client's needs. The project aims to implement the already mentioned real time prediction system and compare it with the previously used prediction model based in the predictions made after the end of the users' session, in other words, a posteriori. Using the developed system several metrics are going to be analyzed, such as precision or recall, in addition to different curves to see the behaviour that they show, such as ROC or PR curve.

Keywords

Digital marketing, prediction, real time, precision, recall, ROC curve.

Agradecimientos

En primer lugar, agradecer a mi tutor Juan Carlos por la paciencia que ha tenido conmigo durante estos meses trabajando en el proyecto. Además, de las correcciones que ha realizado en la memoria y los debates que hemos tenido en el laboratorio.

En segundo lugar, agradecer a la empresa LUCE por la oportunidad que me han ofrecido de poder trabajar en un proyecto como este. En especial a Javier y a Pol que han estado ayudándome y enseñándome en todo momento.

En tercer lugar, agradecer a mis padres y mi hermano por el apoyo que me han dado, ya no durante estos largos meses, si no a lo largo de la carrera y el máster.

En cuarto lugar, agradecer a mis amigos de Cuéllar que han estado apoyándome en todo momento, y comprendían que no podía quedar con ellos todo lo que me gustaría porque tenía que trabajar en el proyecto.

Finalmente, agradecer a mis amigos y compañeros del laboratorio por los ratos tan agradables que hemos vivido en el laboratorio y en los interminables descansos.

Muchas gracias a todos.

Índice

Índice de figuras

Índice de tablas

1 Introducción

En el primer capítulo del documento se introduce brevemente el proyecto desarrollado. En él se responde al porqué del desarrollo de este proyecto en la sección de motivación, posteriormente se enumeran los objetivos finales que se pretenden conseguir en el proyecto, la metodología utilizada y, por último, se da una pequeña descripción de la estructura del documento.

1.1 Motivación

El concepto de comercio ha persistido a lo largo de los años como el intercambio de bienes y servicios entre diferentes partes a cambio de otros bienes y servicios de valor similar [1]. Sin embargo, ha habido innovaciones en torno a la forma de llegar a los usuarios, con una técnica denominada *marketing*, que busca aumentar los beneficios de los vendedores en función de las ofertas que se realizan a los clientes.

Debido a los avances tecnológicos surgió el *marketing* digital, siendo un complemento del *marketing* tradicional. El *marketing* digital ha ayudado a las empresas a desarrollar páginas web para poder vender sus productos por internet. La sobreestimulación a la que fueron sometidos los consumidores condujo a su desinterés, produciendo efectos negativos en las empresas. Para dar una solución a dicho problema, el *marketing* digital recurrió a la hiperpersonalización del mercado publicitario.

En los proyectos de *marketing* digital que se realizan comúnmente el problema esencial radica en la gran ingesta de datos que se almacena en las bases de datos y su posterior procesamiento. En este proyecto no se tiene esa problemática, no se necesitan grandes cantidades de datos ni su procesamiento posterior, sino que el interés está en la respuesta en tiempo real, en la que los datos se recogen en crudo, se realiza un análisis por medio de algoritmos de inteligencia artificial y se dispone de una respuesta inmediata al cliente.

La hiperpersonalización es una técnica de *marketing* en la que las empresas enfocan toda la atención en un cliente en particular, creando así una experiencia personalizada para cada uno de los usuarios de la página web. La mayoría de las empresas deben tener en cuenta y desarrollar esta técnica de *marketing* si pretenden mantener la competitividad y presencia en el mercado. Un gran porcentaje de los consumidores estaría dispuesto a pagar más dinero por una mejor experiencia de cliente, por ello, grandes empresas, como Starbucks, Netflix o Amazon, están trabajando para tener servicios con estas prestaciones [2].

Este proyecto se ha llevado a cabo junto a la empresa *Luce Innovative Technologies*, que se dedica en gran medida al desarrollo y arquitectura de software, analítica, Big Data y *marketing* digital. Este trabajo se ha denominado proyecto *Achiever.*

El desarrollo del proyecto *Achiever* pretende dotar a los consumidores de una experiencia personalizada a partir de diferentes sistemas que puedan predecir la probabilidad de compra de ciertos productos, y con ello poder publicitar al cliente lo que desee o necesite en cada interacción con la página web. Es un proyecto experimental en el que se busca analizar el comportamiento del cliente para poder presentarle el producto más adecuado acorde a sus necesidades, en tiempo real, mejorando así los indicadores de rendimiento clave (KPI- *Key Performance Indicators*) expuestos en la estrategia comercial de la empresa en cuestión.

1.2 Objetivos

El objetivo principal de este proyecto es la creación de un sistema para dotar a la empresa de una hiperpersonalización de aspectos relacionados con las búsquedas que se realizan en las páginas web de comercio electrónico o *e-commerce*.

Los objetivos específicos del proyecto desarrollado son los siguientes:

- Realizar una revisión del estado del arte basada en aspectos relacionados con la hiperpersonalización, además de las métricas que se utilizan en el marketing digital para evaluar la predisposición de compra de los clientes.

- Diseñar y desarrollar una arquitectura para la simulación del comportamiento en tiempo real de las acciones que realizan los clientes.

- Crear un sistema predictivo de hiperpersonalización basado en la recogida de información de diferentes sesiones de clientes procedentes de Google Analytics.

- Evaluar y testear el sistema de tiempo real desarrollado a partir de las métricas del marketing digital, y comparar el sistema con el que desarrolló la empresa LUCE anteriormente.

1.3 Metodología

Para llevar a cabo el proyecto se ha dividido en diferentes fases que se explicarán a continuación.

1. Fase de aprendizaje.

Esta fue la fase preparatoria. En ella se hizo una revisión del estado del arte de diferentes algoritmos de inteligencia artificial que se podían utilizar para llevar a cabo el modelo de predicción de compra. Además, se hizo un estudio de diferentes tecnologías que se pretendían utilizar, sobre todo el uso de la plataforma Google Cloud que fue el servicio donde se recogían los datos de los clientes y de donde se ha partido en el proyecto. Por tanto, fue necesario saber manejar la herramienta con destreza.

2. Fase de estudio del algoritmo de inteligencia artificial

En esta fase se estudió algoritmo utilizado en el modelo de predicción. Se hizo una revisión de los diferentes algoritmos de inteligencia artificial que se utilizan en los modelos de predicción. Este proyecto parte de sistemas ya desarrollados con objetivos distintos a los que se plantean aquí. Sin embargo, se quería aprovechar el trabajo ya realizado, concretamente se quería aprovechar el entrenamiento del algoritmo de predicción existente y comprobar su funcionamiento en un entorno de predicción en tiempo real. Por ello, sólo se realizará el estudio de este para comprender su funcionamiento.

3. Fase de implementación

En esta fase se realizó la implementación del sistema de simulación del tiempo real. Para ello, se crearon dos scripts de Python. El primero de ellos extrae una a una las sesiones de los clientes de Google Analytics, y el segundo es API flask, donde en primer lugar se suben los datos y posteriormente se envian al modelo de predicción de probabilidad de compra. Para realizar esta fase fue necesario tener conocimientos de los lenguajes Python y SQL.

4. Fase de testeo.

La fase de testeo debía analizar el comportamiento de dicho sistema a partir de un conjunto de datos y poder verificar si en realidad se ha acertado en las predicciones realizadas. Para ello, se contó con un conjunto de datos no utilizados previamente en el modelo predictivo. Los resultados obtenidos se compararon con las transacciones reales del cliente y se verificó cómo de eficiente resultaba el algoritmo desarrollado. Para ello, se utilizaron varias métricas de marketing digital: matriz de confusión y curvas ROC y PR, que se definirán en las siguientes secciones detalladamente.

1.4 Estructura de la memoria

La memoria está dividida en cinco capítulos diferentes:

- En el primer capítulo se hará una introducción del proyecto, describiendo las motivaciones para su realización, los objetivos que se pretenden conseguir y la metodología que se ha utilizado para llevarlo a cabo.

- En el segundo capítulo se hará una revisión del estado de arte relacionado con el *marketing* digital y *e-commerce*. Se explicarán las características del *marketing* digital, algunos algoritmos de inteligencia artificial que se pueden utilizar para llevar a cabo el modelo predictivo y, por último, conceptos relacionados con computación en la nube con una comparación de las diferentes plataformas que existen actualmente.

- En el tercer capítulo se describirá el sistema desarrollado. Para ello, se explicarán los dos módulos que se han desarrollado: el módulo de ingesta de datos en tiempo real en el que, además, se explicará el *dataset* utilizado y el módulo de personalización, en el que se explicará cómo se eligió el algoritmo de inteligencia artificial por parte de la empresa LUCE.

- En el cuarto capítulo se expondrán los resultados que se han obtenido tras la realización de los módulos descritos en el tercer capítulo. Para ello se hará una comparación del sistema desarrollado por parte de LUCE con el desarrollado en este proyecto. Además, se hará un análisis de los resultados obtenidos con diferente número de acciones de los clientes, para ver el comportamiento en tiempo real.

- En el quinto capítulo se expondrán las conclusiones del trabajo desarrollado y las posibles líneas futuras.

2 Estado del arte

En el presente capítulo se va a exponer el estado del arte relativo al *marketing* y los avances tanto tecnológicos como sociales que ha habido a lo largo del tiempo hasta llegar al *marketing* digital. Con ello, estaremos en disposición de ver la influencia que estos aspectos han tenido en la predicción de compra en el *e-commerce*. A continuación, se exponen los conceptos anteriormente mencionados: *marketing*, *marketing* digital y *e-commerce*.

La definición de *marketing* presenta diferentes acepciones debido al enfoque que se le quiera dar, el tipo de *marketing* o la época en la que nos encontremos. Por ello, se van a exponer las definiciones más destacadas expuestas por los organismos y asociaciones más relevantes. La RAE la define como: *"Conjunto de principios y prácticas que buscan el aumento del comercio, especialmente de la demanda"* [3]. Sin embargo, esta definición esta más centrada en el acto de la transacción comercial que en el consumidor final de producto. El *Chartered Institute of Marketing (CIM)* concede mayor importancia al cliente y no al comercio como expone: *"el marketing es el proceso de gestión comercial responsable de identificar, anticipar y satisfacer las necesidades y deseos del cliente de forma rentable"* [4]. Como una acepción intermedia nos encontramos la interpretación de la Asociación Americana de Marketing (AMA) en la que da un cierto valor al cliente vinculado a la organización: *"una función de la organización y un conjunto de procesos para crear comunicar y entregar valor a los clientes y manejar las relaciones con estos últimos de manera que beneficien a toda la organización"* [5].

Los avances de la tecnología y la intrusión de internet en el día a día, ha provocado que se tenga todo a nuestro alcance en unos segundos, incluso la realización de un proceso de compra. Este proceso engloba las actividades de búsqueda de información y/o precios de determinados productos. Surge lo que se denomina *marketing* digital definido como un conjunto de acciones llevadas a cabo por medios digitales para promover los productos y empresas [6].

Aunque el *marketing* digital parece un concepto relativamente innovador, surgió en la década de los 90 con la llamada web tradicional o web 1.0 en la que aparecen las primeras plataformas de transferencia de ficheros y las empresas aprovechan para promocionar sus marcas sin la intervención del usuario que solo se dedica a navegar por la web. Es en 2004 cuando surgen las primeras redes sociales, en las que se puede interactuar con el usuario a través de comentarios, en los que manifiestan sus deseos, inquietudes y necesidades [5]. Aquí es donde se produce un gran acercamiento entre las empresas y los consumidores siguiendo con la expansión hasta llegar al año 2009 donde aparece la web 3.0 en la que el *marketing* digital se refiere a las prácticas y métodos de *marketing* que se realizan en internet teniendo en cuenta aspectos relevantes del usuario, como pueden ser ubicación, intereses, etc. En esta fase de desarrollo de la web se trabaja con grandes cantidades de datos, aplicaciones y servicios interoperables, y accesibles desde cualquier dispositivo y lugar gracias al alojamiento en la nube. El siguiente avance es una web 4.0, en la que se obtenga información de los clientes mientras se está navegando. Esos datos serán usados para predecir sus necesidades y deseos. Esto es posible gracias a los algoritmos de inteligencia artificial que se desarrollan en la actualidad [8].

Las empresas en la actualidad buscan un modelo en el que cambie la relación con los clientes englobando las necesidades y expectativas que tienen al contratar un servicio o producto. Con ello, nace lo que se denomina como hiperpersonalización, en la que una empresa maneja los datos de los usuarios y clientes en tiempo real para poder predecir

los productos que le puedan interesar [9]. Muchas empresas están trabajando para llevar a cabo un modelo de hiperpersonalización adecuado, pero solo unas pocas multinacionales parecen tener los recursos necesarios para encabezar este cambio, como por ejemplo Amazon, Spotify, Netflix o Starbucks [2]. Este término nace de las inquietudes de comparación de los productos por parte del usuario antes de realizar una compra. Por tanto, ya no sirve con la realización de una revisión de las cookies del dispositivo, sino que se debe proporcionar una experiencia personalizada a cada usuario. Según un estudio realizado por la agencia Rebold, el 80% de los clientes prefieren comprar a empresas que ofertan una experiencia personalizada [10]. Este nuevo entorno se alcanza [11]:

- Brindando una experiencia en línea individual para cada cliente teniendo en cuenta los datos en tiempo real y los algoritmos de inteligencia artificial durante la interacción con el canal digital.
- Proactividad del sistema para ofrecer productos y servicios deseados o necesarios para el cliente que solo se pueden lograr a partir de la inteligencia artificial.
- Identificación del cliente en tiempo real a partir de la recopilación de todos los históricos del usuario en un determinado momento para poder realizar una predicción adecuada ya sea de gusto del consumidor o de probabilidad de compra de un determinado producto.

Para conseguir llegar a un modelo de hiperpersonalización adecuado se deben tener en cuenta diferentes técnicas. En esta sección se van a postular diferentes apartados para comprender la creación y el funcionamiento de dicho modelo. En primer lugar, se hará una introducción a las características del *marketing* digital para saber los objetivos que debemos de cumplir respecto al modelo. En segundo lugar, se mostrará una clasificación de algoritmos de aprendizaje automático dividiéndose así mismo en aprendizaje automático supervisado y no supervisado. A continuación, se hará una revisión sobre las métricas de evaluación de los diversos algoritmos, para así escoger la opción correcta. Además, se va a mostrar una comparativa de las diferentes plataformas de procesamiento en la nube ya que son las herramientas que se utilizan para el cómputo de grandes conjuntos de datos.

2.1 Características básicas del marketing digital

El *marketing* digital presenta cuatro pilares fundamentales a tener en cuenta para triunfar en cualquier negocio y que pueden ayudar a comprender como funciona el propio *marketing* digital: alcance, compromiso, conversión y retención. [12]

2.1.1 Alcance

El alcance se define como la atracción de visitantes interesados en nuestros productos y servicios para una posible compra o contratación. El objetivo es obtener un mayor número de usuarios cualificados, objetivos e interesados por los productos. Las técnicas más utilizadas para la atracción de clientes son posicionamiento SEO (*Search Engine Optimization*) y SEM (*Search Engine Marketing*). El posicionamiento SEO se define como un conjunto de técnicas que se aplican a las diversas páginas web para mejorar la visibilidad y la posición en los motores de búsqueda de los navegadores de internet. El posicionamiento SEM tiene el mismo objetivo que el posicionamiento SEO, con la diferencia de que SEO se lleva a cabo de forma natural y el posicionamiento SEM se realiza a través de campañas publicitarias en las que el anunciante tiene que reembolsar una cantidad determinada de dinero. [13]

2.1.2 Compromiso

El compromiso abarca las acciones que los usuarios realizan sobre el sitio web al que hayan accedido y que se realice una interacción con el sistema. Estas acciones pueden estar asociadas a múltiples eventos como dejar comentarios, descargar ficheros, realizar lecturas en la página, compartir contenido con las redes sociales, etc. El objetivo es que el usuario no entre y salga de la página en un período corto de tiempo, sino que permanezca en ella lo máximo posible. La técnica utilizada para impulsar el compromiso es la experiencia del usuario (*User Experience*) en la cual se realiza un estudio de como el usuario navega por la página web y obtiene una visión positiva de las necesidades y requisitos del usuario.

2.1.3 Conversión

El término de conversión se refiere a las acciones que un usuario realiza en la página web, pero con la diferencia del compromiso, estas acciones tienen cierto valor para la empresa, ya sea hacerse seguidor de la empresa, comprar algún producto, registrarse en alguna publicación digital, …

2.1.4 Fidelización

La fidelización consiste en la ejecución de diferentes técnicas para que los clientes realicen conversiones repetidas, es decir, que sean leales a la marca. Es un concepto muy importante en el *marketing* digital ya que un usuario que esté satisfecho con los productos y servicios que le proporciona la marca va a realizar más compras y gastar más dinero en ella. Hay diferentes tipos de programas de fidelización dependiendo los productos y servicios que se promocionen como un programa que proporciona recompensas a los clientes para incentivarlos a gastar más, o un programa de sistema de puntos en los que los usuarios pueden obtener determinados productos en función de los puntos que ha recopilado [14].

2.2 Aprendizaje automático en la ciencia de datos

Hoy en día se maneja una inmensa cantidad de datos, tanto estructurados como no estructurados que inundan los mercados. Pero la importancia no radica en la gran cantidad de datos con la que nos encontramos, sino lo que son capaces de hacer las organizaciones con esos datos. Debido a esta cantidad de datos desmesurada nació el término de *Big Data,* que se define como el conjunto de datos que se puede almacenar, procesar y administrar de manera efectiva dentro de un análisis robusto y las herramientas requeridas [15]. Aunque surge la duda de cuál debe ser el tamaño de los datos para que se pueda considerar *Big Data.* Un alto porcentaje de científicos y analistas de datos afirman que la cantidad debe ser desde 30-50 Terabytes hasta varios Petabytes [16].

Debido a la inmensa cantidad de datos que se pueden llegar a almacenar también apareció el término de inteligencia artificial. Este término se utiliza en diferentes campos, según los objetivos, métodos y aplicaciones para los que se deseen utilizar y por ello no existe una definición concreta. No obstante, teniendo en cuenta los objetivos del presente trabajo se podría utilizar la siguiente definición: la habilidad que tienen las máquinas para presentar las capacidades que tienen los seres humanos, como el aprendizaje.

Dentro de la inteligencia artificial nos encontramos con una rama llamada aprendizaje automático la cual proporciona a los ordenadores la capacidad de aprender usando patrones de datos para poder generar un modelo matemático. Los patrones se generan a partir de los algoritmos, y con ellos se forma el modelo matemático para realizar las predicciones. A medida que aumenten el conjunto de datos de entrada al modelo y las

iteraciones realizadas, los resultados serán más precisos; al igual que pasa con las actividades realizadas por un ser humano, mejora con la práctica. Dependiendo de la necesidad del problema a resolver, los aspectos que afectan a la toma de decisiones y el entorno en el que se va a desarrollar destacan dos tipos de aprendizaje automático: aprendizaje supervisado y aprendizaje no supervisado. [17]

2.2.1 Aprendizaje supervisado

El aprendizaje supervisado tiene como objetivo aprender a partir de un conjunto de datos etiquetados, es decir, categorizados o agrupados, que permita conformar un modelo de predicción. Con el grupo de datos de entrada, llamados datos de entrenamiento se van realizando los ajustes del modelo predictivo. El algoritmo de aprendizaje va aprendiendo la clasificación de las diferentes muestras a partir de sus propias etiquetas y del resultado del modelo, para así poder corregir los errores en las estimaciones de los resultados. En la Figura 1 se muestra el diagrama de flujo de un modelo de aprendizaje supervisado, en el que el algoritmo de aprendizaje automático va aprendiendo gracias a los vectores de características, proporcionados por textos de entrenamiento, documentos, imágenes, etc. Además de los vectores de características se proporcionan las etiquetas para conformar el modelo predictivo. Una vez conformado el modelo, se realiza la fase de testeo en la que se lanzan nuevos datos con las mismas características que los datos de entrenamiento y así poder generar la predicción de la etiqueta esperada [18].

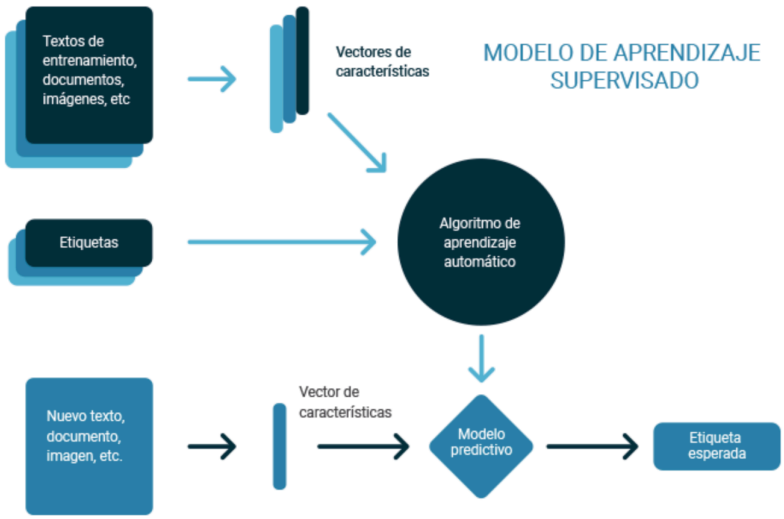


Figura 1- Diagrama de flujo de un modelo de aprendizaje supervisado [18].

Dentro del aprendizaje supervisado se engloban numerosos algoritmos de aprendizaje automático como los que se van a exponer a continuación: *linear regression, logistic regresion, support vector machine, decision trees* y *random forest*.

2.2.1.1 Linear regression

El algoritmo *linear regression* o regresión lineal en castellano es uno de los algoritmos más utilizados en aprendizaje supervisado dada su sencillez. Consiste en modelar una variable dependiente Y como combinación lineal de n variables independientes X_i y un término libre γ. La expresión de la variable dependiente es la que se muestra en la Ecuación 1.

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_n X_n + \gamma$$

Ecuación 1- Expresión del algoritmo de regresión lineal simple.

Los valores de β_i son constantes y son los términos que deben optimizarse a partir de los datos de entrenamiento. Se verán maximizados cuando el error entre el estimador y el dato de entrenamiento sea mínimo.

Para el caso del modelo lineal simple solo se consideran los términos β_0 y β_1, los cuales serán la ordenada en el origen y la pendiente de la recta respectivamente. Un ejemplo del modelo de regresión lineal se puede ver en la Figura 2, donde los puntos azules son los datos del entrenamiento y la línea roja la aproximación lineal que se ha obtenido tras poner en marcha el modelo.

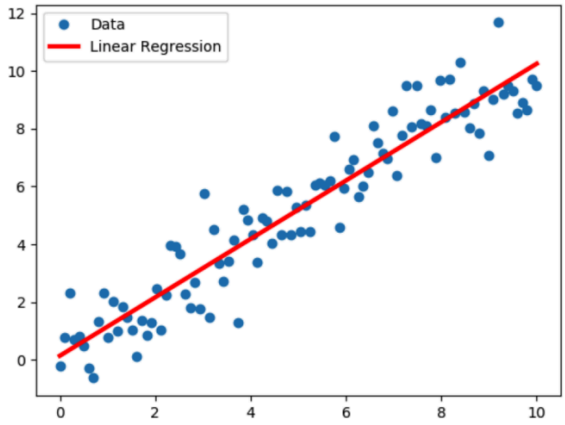


Figura 2- Regresión lineal de un conjunto de datos de entrenamiento [19].

2.2.1.2 Logistic Regression

El algoritmo de regresión logística utiliza *logits* para realizar las predicciones de la clase objetivo. Utiliza una función como si fuese una caja negra, llamada función *Softmax*, en la que relaciona la variable dependiente que se quiere predecir con las variables independientes que son las demás variables que se utilizar para llevar a cabo las predicciones.

Para entender este algoritmo debemos entender el ratio de probabilidad, como el ratio de que ocurra un evento. Se puede expresar con la siguiente expresión $\frac{p}{1-p}$, donde p es la probabilidad de que ocurra dicho evento. Podemos utilizar logaritmos ya que es la función que caracteriza la regresión logística quedando así $logit(p) = \log\left(\frac{p}{1-p}\right) = \log(p) - \log(1-p)$. Estos *logits* son el resultado de las variables de entrada (x) por los pesos (w) de cada una de ellas como se puede ver en la Ecuación 2.

$$\text{logit}(p) = w_0 x_0 + w_1 x_1 \dots w_n x_n = \sum_{i=0}^{n} w_i x_i = \boldsymbol{w^T x}$$

Ecuación 2- Expresión de logit para el algoritmo de regresión logística.

Estas serán las puntuaciones obtenidas que pasarán a la función *Softmax* para calcular la probabilidad de cada clase proporcionada. En la Figura 3 se puede ver un diagrama con los pasos que se dan para la aplicación del algoritmo.

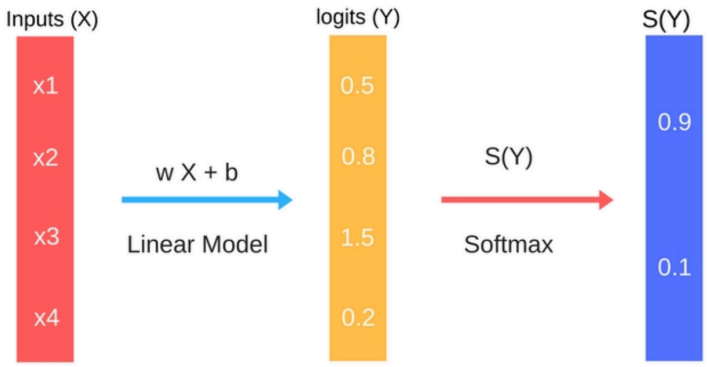


Figura 3- Diagrama de clasificación del algoritmo de regresión logística

2.2.1.3 Support Vector Machine (SVM)

El algoritmo SVM se caracteriza por ser un algoritmo de clasificación de aprendizaje automático y no de regresión como en los casos anteriores. Dicho algoritmo no presenta un fundamento matemático destacado, se basa en el proceso de entrenamiento de una función que maximice la distancia entre dos agrupaciones o etiquetas para así poder crear un hiperplano que las distinga. En la Figura 4 se muestra la representación de un ejemplo de funcionamiento del algoritmo SVM. En él nos encontramos con dos categorías (categoría A y B) y un hiperplano representado por una línea recta que separa los dos conjuntos. Así, se obtiene una clasificación de las distintas categorías.

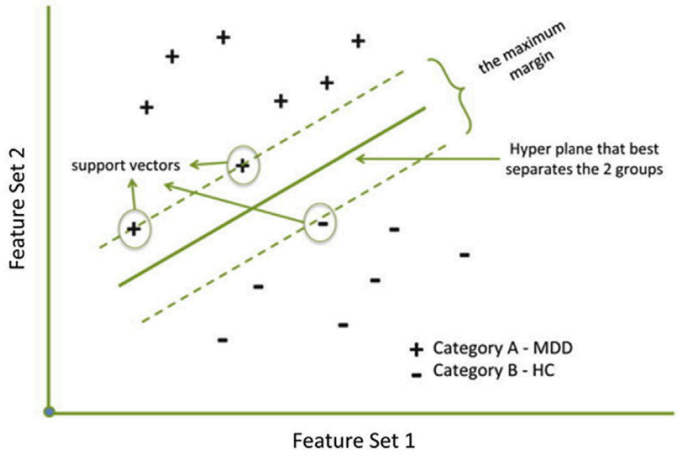


Figura 4- Representación del algoritmo SVM mediante support vectors separados por el hiperplano [20].

Puede haber varios límites de decisión para categorizar las clases en un espacio n-dimensional, aunque debe haber un plano que maximice la clasificación y suele ser en función de la distancia entre los dos conjuntos.

2.2.1.4 Decision trees

El algoritmo *decision trees* o árbol de decisión en castellano es un algoritmo enfocado tanto para la clasificación supervisada como para la regresión. La idea es semejante a un árbol ordinario en el que tenemos una raíz, y diferentes nodos, los cuales se comunican con ramas, hasta llegar a las hojas. En esta estructura los nodos serán pequeños conjuntos de datos, las ramas las reglas de decisión y las hojas las clases o categorías con las que estamos trabajando [21].

En cuanto a la estructura que presenta el árbol de decisión se puede ver en la Figura 5 donde aparecen las ramas, las hojas y los nodos de decisión descritos anteriormente.

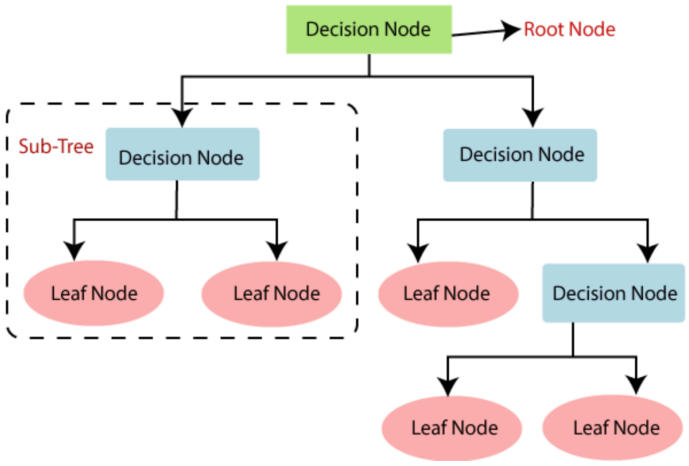


Figura 5- Diagrama de flujo de la representación de un árbol de decisión [22].

La problemática de este algoritmo surge en cómo crear el árbol de decisión, ya que se deben elegir y distribuir los atributos en los nodos. Para ello, existe una técnica denominada ASM o selección de atributos. Dentro de la misma podemos usar dos métodos para llevar a cabo el árbol: ganancia de información e índice de Gini.

Ganancia de información: es una métrica que proporciona cuánto de separados están los conjuntos una vez que han pasado por los nodos de decisión. El caso ideal sería que los conjuntos de diferentes nodos sean lo más heterogéneos posibles y eso se consigue con una ganancia de información grande.

Índice de Gini: es un coeficiente entre 0 y 1, el cual nos muestra la homogeneidad de un nodo, donde el valor 0 nos indica que la agrupación de los datos son puros y al ir incrementando el valor hasta 1, incrementa su heterogeneidad. En la Ecuación 3 se muestra la expresión del índice de Gini para un cierto conjunto D dada la probabilidad de que la categoría p_i se encuentre en el conjunto D [22].

$$Gin(D) = 1 - \sum_{i=1}^{n} p_i^2$$

Ecuación 3- Expresión del índice de Gini para un conjunto D con una probabilidad p de que esté dentro del conjunto

2.2.1.5 Random forest

El algoritmo de *random forest* es un algoritmo creado a partir de múltiples árboles de decisión, lo que se conoce como un algoritmo ensamblado. La principal problemática que presentan los árboles de decisión es el *overfitting,* problemática que surge cuando el modelo se ha optimizado en exceso para un determinado conjunto de datos de entrenamiento provocando un error en el conjunto de prueba si los datos distan lo más mínimo del conjunto de entrenamiento. Por tanto, para resolver este problema se crean varios árboles de decisión en el que cada uno realiza una predicción individual, y a través de un sistema de votos se elige la opción más votada. La estructura definida anteriormente se muestra esquematizada en la Figura 6, en la cual, se distinguen cada uno de los árboles de decisión, sus respectivas predicciones y puestas en común para su elección final.

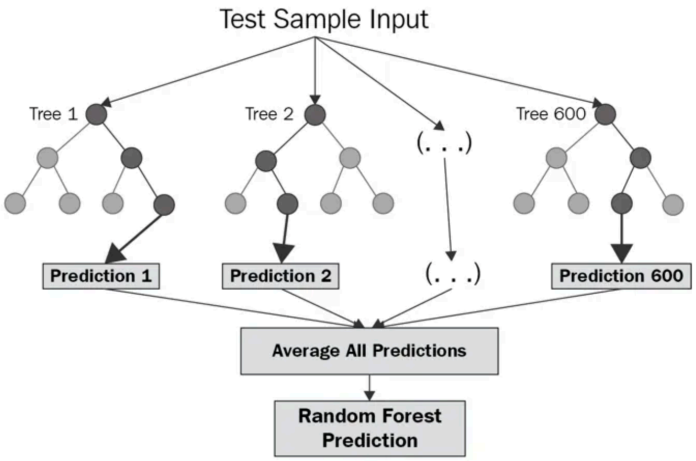


Figura 6- Esquema del algoritmo de random forest [24].

En el caso de utilizar este algoritmo hay que tener en cuenta ciertos parámetros principales del algoritmo: el número de árboles de decisión y la profundidad de dichos árboles. El número de árboles de decisión por lo general suele ser alto, dado que cuantos más árboles haya, disminuimos los errores, pero en contraposición, no puede ser extremadamente grande ya que hará que el modelo sea muy lento, por tanto, nos encontramos frente a un compromiso entre veracidad y velocidad. La profundidad del árbol viene definida por el límite de divisiones que se llegan a hacer en un solo árbol [25].

Como conclusión de *random forest* hay que destacar que posee mejor rendimiento que los árboles de decisión ya que realiza una compensación de errores de las predicciones de cada uno de los árboles. Sin embargo, también presenta alguna desventaja frente a otros algoritmos de aprendizaje automático. Ofrecen una gran inestabilidad cuando se producen pequeños cambios, ya que se deben realizar modificaciones en la estructura del árbol. Este defecto se puede solventar con dos mecanismos denominados *bagging* y *boosting*. En el caso de *bagging* realiza un aprendizaje paralelo y en el caso de *boosting* lo realiza secuencialmente, obteniendo pesos diferentes en cada iteración [26].

2.2.2 Aprendizaje no supervisado

El aprendizaje no supervisado permite aprender a partir de un conjunto de datos sin etiquetar. Por tanto, debemos manejar datos sin estructura y sin información previa que nos ayude a clasificar o categorizar los datos. Así este tipo de aprendizaje intenta crear subgrupos del conjunto de datos viendo las similitudes de sus características para así crear lo que se denomina como *clusters*. Hay que tener en cuenta diferentes parámetros para crear los *clusters*, en primer lugar, el número de agrupaciones que vamos a realizar en todo el conjunto de datos y, en segundo lugar, la elección de la posición del centroide (punto donde empieza el *cluster*). Dependiendo del conjunto de datos que se tengan, estos parámetros cambiarán, por tanto, se deberá hacer un muestreo de muchas iteraciones para ver cuál es la opción óptima.

He de destacar que el conjunto de datos con el que se va a trabajar en este proyecto está etiquetado, por lo que no se van a utilizar algoritmos de aprendizaje no supervisado. Por esta razón, solo se va a describir un algoritmo de aprendizaje no supervisado. Este algoritmo se denomina *K-means* y se ha elegido para explicar porque es el más utilizado y simple que se ha encontrado.

2.2.2.1 K-means

Es un tipo de algoritmo de aprendizaje no supervisado que agrupa los datos en K conjuntos dependiendo de las características de los datos. El algoritmo se construye a partir de los siguientes pasos:

1. Definición del número de *clusters* o agrupaciones (K).
2. Elección de la posición de los K centroides aleatoriamente.
3. Asignación de cada objeto de dato a su centroide más cercano utilizando una métrica de distancia.
4. Actualización de centroides realizando un promedio de los puntos del *cluster*.

Los puntos 3 y 4 se evalúan tantas veces como sea necesario hasta que los centroides se estabilicen [27].

2.2.3 Métricas de un algoritmo

Hasta ahora hemos realizado una descripción de diferentes algoritmos utilizados para la clasificación de conjuntos de datos, tanto etiquetados como no etiquetados. En este apartado se van a analizar las métricas de rendimiento que existen para la evaluación de estos algoritmos. Estas métricas nos ayudarán en nuestro proyecto para ver el algoritmo que más nos conviene en función de los datos de entrada.

Antes de comenzar con la definición de las métricas es conveniente realizar un análisis de conceptos previos para así saber de dónde se extraen. Para lo cual, es necesario explicar qué es la matriz de confusión, herramienta utilizada para evaluar el rendimiento de dichos algoritmos. Para poder utilizar esta matriz se deben dividir los datos de entrenamiento en dos agrupaciones diferentes: un alto porcentaje de los datos para realizar el entrenamiento (80%) y el resto para realizar el testeo (20%). La matriz de confusión es una matriz cuadrada en la que se realiza un recuento de los valores predichos frente a los valores reales. En la Tabla 1 se muestra la matriz de confusión las filas representan las observaciones y las columnas los valores predichos. Como se puede ver en la tabla inferior los valores de la diagonal principal se encuentran bien clasificados y los que están fuera de ella erróneamente clasificados [28].

		Predicción	
		Positivos	**Negativos**
Observaciones	**Positivos**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos positivos (FP)	Verdaderos Negativos (VN)

Tabla 1- Matriz de confusión genérica

A partir de esta matriz de confusión se pueden definir las diferentes combinaciones entre observaciones y predicciones como las que siguen:

- Verdaderos Positivos (VP): es la cantidad de positivos que el modelo clasificó como positivos.
- Falsos Negativos (FN): es la cantidad de positivos que el modelo clasificó como negativos.

- Falsos Positivos (FP): es la cantidad de negativos que el modelo clasificó como positivos.
- Verdaderos Negativos (VN): es la cantidad de negativos que el modelo clasificó como negativos.

A partir de estos cuatro parámetros de la matriz de confusión podemos obtener las métricas de rendimiento para conseguir más información sobre el algoritmo que se va a escoger. Las métricas estudiadas son las siguientes: precisión, exactitud, sensibilidad y F1 score [29].

2.2.3.1 Precisión

Es una métrica que evalúa solamente los valores verdaderos de la clase que se quiere obtener, es decir, de los positivos. En la Figura 7 nos encontramos con la matriz de confusión con las elecciones que se han realizado para el cálculo de la precisión, por un lado, los verdaderos positivos y por otro las predicciones positivas totales (tanto verdaderos como falsos). El cálculo viene definido por la expresión que se muestra en la Ecuación 4.

		Predicción	
		Positivos	**Negativos**
Observaciones	**Positivos**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos positivos (FP)	Verdaderos Negativos (VN)

Figura 7- Matriz de confusión con la elección para cálculo de la precisión.

$$Precisión = \frac{VP}{VP + FP}$$

Ecuación 4- Expresión para el cálculo de la precisión.

Esta métrica nos puede ayudar a saber si el modelo de predicción presenta muchos falsos positivos. El caso ideal sería que no hubiese falsos positivos y la precisión tenga un valor igual a 1, es decir que todas las predicciones positivas sean valores reales positivos.

2.2.3.2 Exactitud

Métrica que nos indica el número de predicciones acertadas que ha realizado el modelo frente al total de predicciones. Esta métrica no nos indica que el modelo funcione correctamente si la exactitud es cercana a 1, ya que podemos encontrarnos en la situación de que el número de verdaderos negativos sea muy superior al de verdaderos positivos y, en esa situación, aunque los falsos positivos sean escasos pueden ser significativos frente a los verdaderos positivos. En la Figura 8 se muestra una representación de la matriz de confusión. En ella se puede ver que se clasifican por un lado las predicciones verdaderas (tanto positivos como negativos) y por otro lado todas las predicciones. La expresión para el cálculo de la exactitud viene definida en la Ecuación 5, a partir de las agrupaciones que se han realizado.

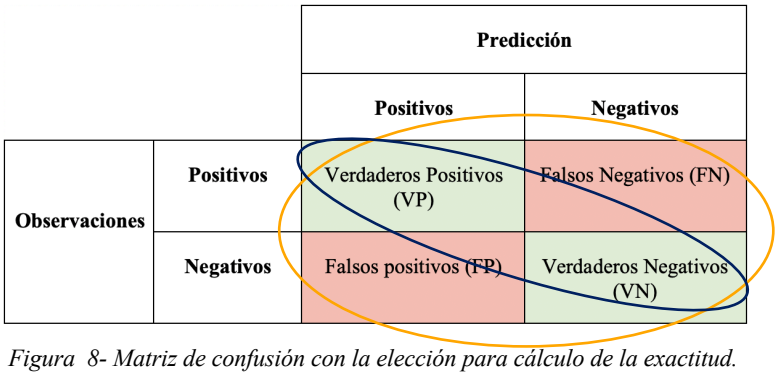


Figura 8- Matriz de confusión con la elección para cálculo de la exactitud.

$$Exactitud = \frac{VP + VN}{VP + VN + FP + FN}$$

Ecuación 5- Expresión para el cálculo de la exactitud.

Esta métrica es muy útil para tener una visión general del comportamiento del sistema en cuanto a las predicciones acertadas, ya que se evalúan las verdaderas predicciones frente a las predicciones totales.

2.2.3.3 Especificidad

Métrica que mide la tasa de falsos positivos. Esta tasa se calcula como el número de falsos positivos entre el número total de observaciones negativas, como se muestra en la Figura 9 con las elecciones pertinentes para llevar a cabo el cálculo. La expresión que determina esta métrica se parecía en la Ecuación 6.

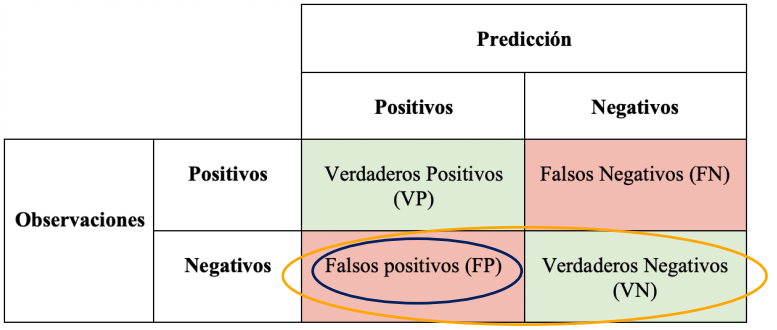


Figura 9- Matriz de confusión con la elección para el cálculo de la especificidad.

$$Especificidad = \frac{FP}{VN + FP}$$

Ecuación 6- Expresión para el cálculo de la especificidad.

Dentro del ámbito de la predicción de compra, la especificidad se puede definir como los casos que se ha predicho que se iba a realizar la compra, pero en realidad no se ha realizado. Si este valor es muy alto se puede argumentar que el umbral de decisión que se ha elegido no es el óptimo, ya que el número de falsos positivos es muy elevado.

2.2.3.4 Sensibilidad (recall)

Métrica que mide la tasa de verdaderos positivos. Es la proporción verdaderos positivos frente al total de observaciones positivas. En la Figura 10 se muestra la matriz de confusión definida anteriormente. En ella se ha seleccionado por un lado los verdaderos positivos y por otro el conjunto total de observaciones positivas para realizar el cálculo de la tasa de verdaderos positivos como se observa en la Ecuación 7.Ecuación 7- Expresión para el cálculo de la sensibilidad.

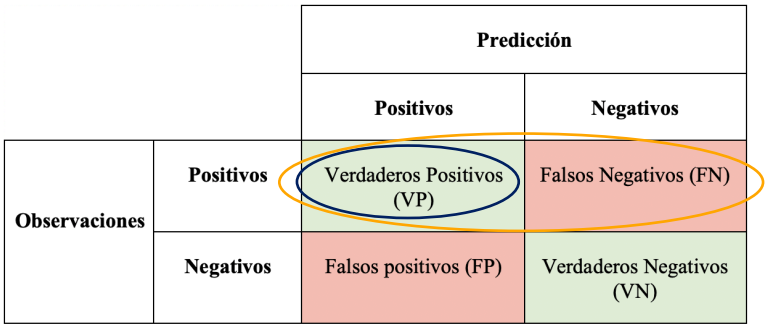


Figura 10- Matriz de confusión con la elección para el cálculo de la sensibilidad

$$Sensibilidad = \frac{VP}{VP + FN}$$

Ecuación 7- Expresión para el cálculo de la sensibilidad.

Como en el caso anterior, en el ámbito de la predicción de compra se puede definir como el número de casos que se ha predicho que se iba a comprar y que realmente se ha comprado. Si este valor es muy alto quiere decir que para el umbral de decisión utilizado el modelo de predicción funciona correctamente.

2.2.3.5 Índice F1 score

El valor F1 se utiliza para obtener una métrica más al combinar la precisión y sensibilidad. Este valor es más práctico a la hora de poder evaluar el rendimiento de un algoritmo que con las dos métricas por separado. El índice F1 se calcula como la media armónica entre las dos métricas citadas como se muestra en la Ecuación 8.

$$F1 = 2 \cdot \frac{precisión \cdot sensibilidad}{precisión + sensibilidad}$$

Ecuación 8- Expresión para el cálculo del valor F1.

Esta expresión se usa únicamente si la precisión y la sensibilidad tienen el mismo peso, es decir, no le damos más importancia a una métrica que a otra, pero en la práctica pocas veces nos vamos a encontrar en esta situación. En la Ecuación 9 se muestra la expresión generalizada del índice F_β.

$$F_\beta = (1 + \beta^2) \frac{precisión \cdot sensibilidad}{\beta^2 \cdot precisión + sensibilidad}$$

Ecuación 9- Expresión genérica para el cálculo del valor F

2.2.3.6 Umbrales de decisión, curvas ROC y PR

Como interpretación gráfica de algunas de las métricas mencionadas, surgió la curva ROC (Receiver Operating Characteristic) y el área bajo la curva AUC. La curva ROC se representa a partir de la tasa de verdaderos positivos, frente a la tasa de falsos positivos al realizar un barrido del umbral de decisión desde 0 hasta 1. Por tanto, se debe definir un umbral de clasificación para dividir las categorías en positivos y negativos, si la predicción está por encima del umbral, la categoría es aceptada, y por el contrario si está por debajo la categoría es rechazada [29]. Como resultado se puede ver la curva ROC en la Figura 11. En ella se aprecia que el clasificador perfecto resulta cuando la tasa de verdaderos positivos es igual a 1 y la tasa de falsos positivos es igual a 0. Sin embargo, esto solo es observable en un modelo predictivo teórico no reproducible en la realidad,

quedando el clasificador aproximado a la línea roja y a la línea azul representada en el caso de que el modelo sea buen clasificador.

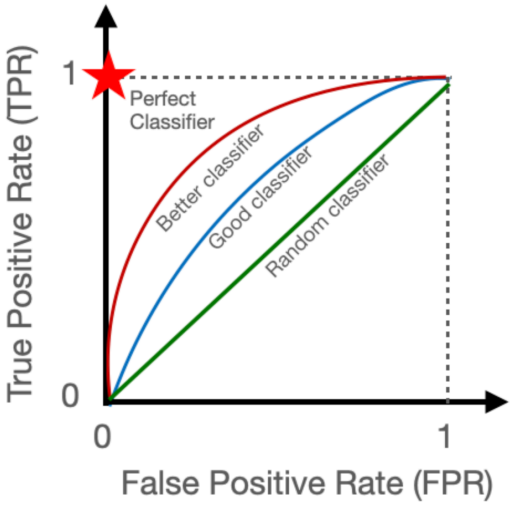


Figura 11- Representación de la curva Roc, como la tasa de verdaderos positivos frente a la tasa de falsos positivos [29].

El parámetro AUC, área bajo la curva se puede calcular realizando la integral bajo la curva ROC. Así, cuanto mayor es el valor AUC mejor es el rendimiento del clasificador en cuestión, coincidiendo con las conclusiones dadas en el análisis de la curva ROC.

Otra de las curvas que pueden analizar es la curva PR, a partir de dos métricas que están relacionadas, la precisión y el *recall*. Estas son métricas opuestas en el sentido de que generalmente, si entrenamos el modelo para que aumente la precisión, disminuirá el *recall* y viceversa [30]. Como ejemplo podemos ver la gráfica mostrada en la Figura 12 en la que se muestra la precisión y el *recall* en función de un umbral de decisión, si se aumenta el valor del umbral, la precisión decrece y el *recall* crece. Por tanto, nos encontramos ante el compromiso de qué es mejor aumentar si la precisión o el *recall*. Pues bien, depende el caso con el que se esté trabajando nos interesará aumentar uno u otro variando así el umbral de decisión.

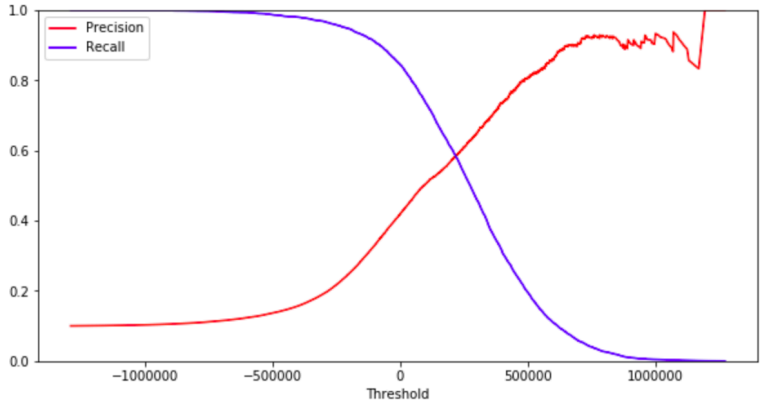


Figura 12- Representación de la curva PR en función del umbral de decisión [30].

La representación que da como resultado la combinación de estas dos métricas es la curva PR que se muestra como ejemplo en la Figura 13, en la que se puede ver la diferencia entre el rendimiento de dos algoritmos. Como se ha comentado interesa un alto valor de

precisión y un alto valor de *recall*, por tanto, el algoritmo que presenta mejor rendimiento frente a estos factores es el *Algorithm 2,* marcado con la línea roja.

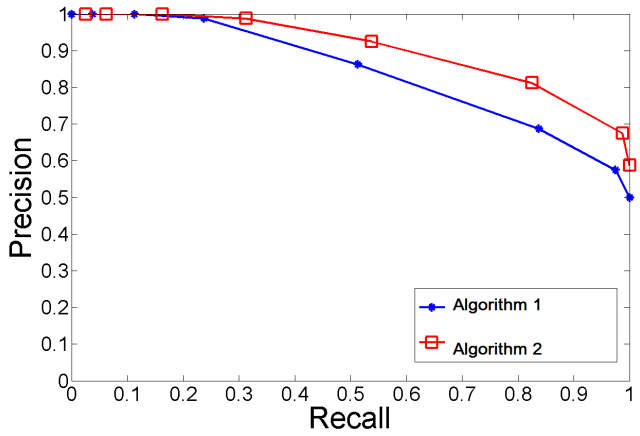


Figura 13- Representación de la curva PR, enfrentando la precisión y la sensibilidad

Como se ha visto en las diferentes gráficas un valor muy importante a tener en cuenta es el umbral de decisión que se escoge para comparar con la probabilidad de que ocurra un suceso. Este valor hará que el modelo de decisión presente o no los resultados esperados.

Las clases positivas y negativas se pueden definir como dos distribuciones gaussianas como se muestran en la Figura 14. Para que el modelo funcionase sin errores, es decir, sin falsos positivos y sin falsos negativos, no se tendrían que cortar las dos curvas de densidad de probabilidad. Sin embargo, rara vez nos encontramos con esta situación y vamos a tener fallos.

Como se ha expuesto en las gráficas anteriores en función de los falsos positivos y negativos se definen métricas como el *recall* y la precisión que pueden mejorar el modelo. Si queremos una proporción equitativa entre falsos positivos y negativos se debe elegir el umbral de decisión en el punto donde se corten dichas gráficas como se muestra en la figura.

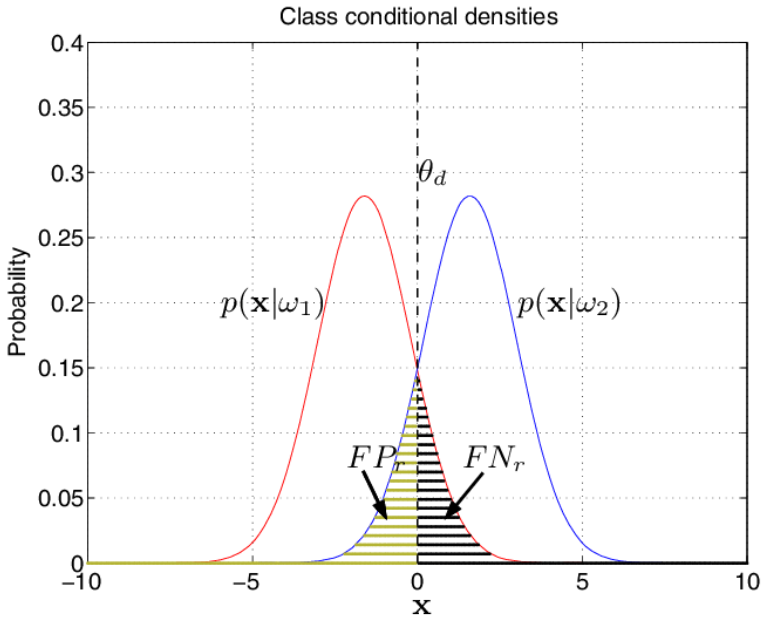


Figura 14- Gráfica que representa la función de densidad de probabilidad de las clases positivas y negativas como dos distribuciones gaussianas.

Las métricas descritas en esta sección son los principales parámetros que se utilizan para la evaluación de algoritmos de *Machine Learning*. Por ello, son métricas que se han utilizado a lo largo del proyecto para observar los resultados que se han obtenido, y ver si el modelo de predicción era el adecuado.

2.3 Computación en la nube

En la actualidad, el aumento de la carga de datos trasmitidos y procesados ha condicionado la forma en la que la sociedad, las corporaciones y los individuos interactúan con los datos. Como consecuencia de esta presión sobre los recursos se desarrolló la computación en la nube o *Cloud Computing.* Este término tiene diversas definiciones según el autor. En este proyecto se ha definido como un sistema de servidores remotos que ofrecen recursos informáticos bajo demanda, ya sean aplicaciones, servicios, almacenamiento, redes y servidores, a los usuarios finales por un precio conforme a las necesidades requeridas. En concreto, la nube de computación presenta tres modelos diferentes dependiendo de las necesidades que tenga el usuario o la empresa: SaaS (*Software as a Service*), IaaS (*Infraestructure as a Service*) y PaaS (*Platform as a Service*) que detallaremos a continuación [32].

2.3.1 Modelos de computación en la nube

***SaaS* (*Software as a Service*):** modelo de computación en la nube destinado a ofrecer aplicaciones a los usuarios finales a través de internet. Así, los usuarios pueden acceder a la misma aplicación desde diferentes máquinas o dispositivos mediante un buscador o aplicaciones. Las soluciones SaaS se ofrecen a los clientes a través de pagos por suscripción.

***IaaS* (*Infraestructure as a Service*):** modelo de computación en la nube el cual provee y gestiona recursos de computación en internet, como almacenamiento, servidores, redes o virtualización. Estas infraestructuras se ofrecen como centros de datos de grandes dimensiones y con un coste relativamente bajo en comparación con la inversión de capital que se tendría que realizar si se requiere del hardware y equipamiento tecnológico necesario.

***PaaS* (*Platform as a Service*):** modelo de computación en la nube enfocado para desarrolladores de las aplicaciones. Se ofrece un entorno propicio para realizar un código de calidad y aplicaciones personalizadas. Los desarrolladores no tienen la preocupación por el almacenamiento o sistemas operativos ya que se provee de servidores los cuales se pueden configurar según las necesidades de la aplicación.

2.3.2 Tipos de nubes de computación

Las nubes de computación presentan ciertas similitudes dado que todas comparten, extraen y agrupan los recursos informáticos que presenta la red. Además, todas están formadas por los mismos sistemas tecnológicos, ya que poseen un sistema operativo, una plataforma de gestión e interfaces de programación para las aplicaciones. Sin embargo, se puede hacer una clasificación de estas en función de la propiedad y ubicación. La distinción realizada son nubes públicas, privadas e híbridas [33].

Nubes públicas: la infraestructura de la nube se proporciona para el uso público. Se suelen crear a partir de infraestructuras ajenas al usuario final. Algunos proveedores de nubes públicas más conocidos son Google Cloud, Amazon Web Servicies (AWS), Microsoft Azure o IBM Cloud. Debido al aumento de cantidad de datos que se ejecutan en las empresas, los proveedores de las nubes han comenzado a ofrecer los servicios dentro de las mismas.

Nubes privadas: la infraestructura de la nube es proporcionada para un solo usuario o grupo de usuarios finales restringiendo el acceso a los demás usuarios, como puede ser una organización con múltiples usuarios. La entidad que opera y gestiona la nube puede ser la misma entidad que proporciona los servicios, la que consume los servicios o puede haber un acuerdo para realizar una gestión como combinación de ambas.

Nubes híbridas: la infraestructura de la nube es una composición de dos o más nubes diferentes de los modelos mencionados anteriormente. Permite el equilibrio de carga entre diferentes nubes. Típicamente surgen como aumento de los servicios de una nube privada, es decir, la nube privada llega al límite de carga, se utiliza una nube pública para llevar a cabo estas necesidades.

2.3.3 Plataformas de computación en la nube

En este apartado se va a realizar una revisión sobre las grandes plataformas de computación en la nube que existen a nivel mundial. Un estudio realizado en el primer trimestre de 2021 examina la cuota de mercado de 500 empresas en 49 países del mundo. En la Figura 15 se muestran las corporaciones que más cuota de mercado han tenido en el período de tiempo mencionado.

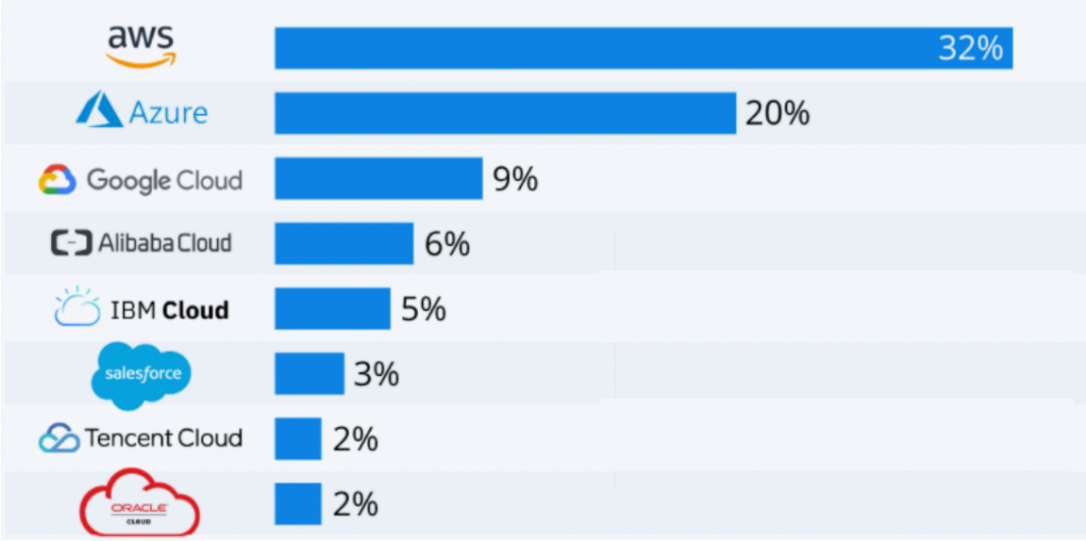


Figura 15- Clasificación de la cuota de mercado de la nube en el primer cuatrimestre de 2021 [34].

Según los datos estadísticos Amazon Web Services (AWS) lidera la clasificación con un 32% de la cuota de mercado mundial de la nube englobando los tres modelos de computación que se mencionaron en el punto 2.3.1. Según la plataforma de análisis de gastos de empresas *parkmycloud,* AWS generó 13,5 mil millones de dólares en este primer cuatrimestre. En segunda posición con una cuota de mercado mundial del 20%, se encuentra Microsoft Azure, el cual dio un gran paso hacia delante respecto al trimestre anterior, aumentando sus ingresos un 50% llegando a alcanzar ingresos de hasta 15,1 mil millones de dólares incluyendo los servicios y servidores proporcionados por la nube. En tercer lugar, se posiciona Google Cloud Platform con una cuota del 9% del mercado mundial y llegando a obtener unos ingresos de 4,047 mil millones de dólares lo que supone un gran aumento y crecimiento de la empresa en este ámbito. Las demás plataformas también han tenido un gran crecimiento en el primer trimestre de 2021, sin embargo, nos vamos a centrar en analizar las características de las tres principales plataformas de computación en la nube: AWS, Azure y Google Cloud [34].

2.3.3.1 Comparativa AWS, Azure y Google Cloud

En este subapartado se va a realizar una comparativa de las principales plataformas en la nube a nivel mundial, como son AWS, Microsoft Azure y Google Cloud. Las principales características por analizar son computación, red, almacenamiento y seguridad [35].

Características de computación

Las características de computación son la base sobre la que se construye la nube en cuestión. Por ello, la elección de los recursos informáticos es clave para el rendimiento y velocidad de la plataforma. Para realizar una comparación sobre la capacidad de computación de cada una de las alternativas nos hemos basado en las máquinas virtuales, ya que es el punto central sobre el que se construye todo sistema informático. En cuanto a la oferta que presentan cada uno de los proveedores nos encontramos con *Amazon Elastic Compute Cloud* en el caso de AWS, *Azure Virtual Machines* en el caso de Azure, y *Compute Engine* en el caso de Google Cloud.

En cuanto a las instancias que se despliegan en las máquinas virtuales cabe destacar varias similitudes entre las tres plataformas:

- Utilización de imágenes almacenadas en el disco para crear nuevas instancias.
- Administración de las instancias, desde iniciarlas, pararlas o borrarlas. Además, de poder realizar un etiquetado en todas ellas.
- Capacidad de poder instalar diversos sistemas operativos en las instancias.

Dentro del ámbito de las instancias se puede destacar como principal diferencia que el acceso a las máquinas virtuales a través del terminal mediante SSH requiere claves SSH propias tanto en AWS como en Azure. Sin embargo, para la plataforma de Google Cloud no hace falta incluir las claves, sino que podemos crear tantas claves como instancias tengamos o necesitemos, además, tampoco es necesario almacenar las claves SSH en la máquina local ya que presenta un terminal SSH basado en navegador llamado *Google Cloud Console*.

Una de las características más importantes de las máquinas virtuales es el número de CPUs que puede llegar a soportar y la memoria RAM que presenta dicha máquina. En este aspecto la plataforma que soporta mayor número de CPUs es AWS con un total de 448, seguido de las otras dos plataformas con un total de 416. El tamaño de memoria RAM también lo lidera AWS con 24.576 GB, seguido de Google Cloud y Azure con 11.776 GB y 11.400 GB, respectivamente.

Características de la red

Las tres plataformas presentadas han desarrollado una infraestructura de red global. Cada una de ellas ha desplegado sus propias redes, en las cuales interconecta diversas nubes para prestar al usuario final una alta velocidad de datos, con gran tolerancia a fallos y baja latencia.

Los centros de datos de las diferentes plataformas se encuentran en diversas localizaciones geográficas. En el caso de AWS nos encontramos con gran cantidad de centros de datos, llegando a 245 países y territorios a lo largo del panorama internacional y con aspiraciones a crear nuevos centros en otros territorios. En cuanto a las dos plataformas restantes también cuentan con numerosos centros de datos esparcidos a lo largo del globo terráqueo, pero no llegando a los números que presenta AWS. La plataforma de Google Cloud la conforman centros de datos repartidos por 200 países y Microsoft Azure cuenta con 170 puntos de red. Se aprecia que todas ellas presentan

puntos de acceso en casi todos los puntos a nivel internacional y siguen aumentando día a día.

Otro factor que no se puede olvidar comparar es la latencia de la red de la nube, ya que interesa que sea lo más pequeño posible. Un estudio realizado por *Cockroach Labs* en 2021, midió la latencia de las tres plataformas, obteniendo un resultado favorable para AWS, seguido de Azure y por último Google Cloud. Este estudio se realizó bajo ciertas condiciones, ya que podría variar la latencia dependiendo la zona en la que se encuentre el cliente y donde esté ubicada la instancia de la máquina virtual asociada.

Características de almacenamiento

En este apartado se van a definir las características de almacenamiento de las plataformas citadas anteriormente. Las opciones de almacenamiento que podemos encontrar se dividen en 4 grandes tipos.

- Almacenamiento de objetos distribuidos. Definido como el proceso de almacenar objetos de tipo *blob*. Cada uno de los objetos tiene una clave única y un registro con las propiedades esenciales del conjunto de datos, como puede ser la fecha de creación, de modificación, etc.
- Almacenamiento en bloque. Es la integración de un disco virtual a una máquina virtual dentro de la nube. La integración de los discos se puede realizar de dos maneras, mediante discos que se conectan a la red, es decir, volúmenes que se conectan a la máquina virtual, y mediante discos conectados a la máquina local (física).
- Almacenamiento de ficheros. Las plataformas ofrecen un servicio de almacenamiento de ficheros integrado en la nube capaz de crear, gestionar y configurar sistemas de ficheros rápidamente. El protocolo utilizado para la configuración de los sistemas de ficheros es NFSv4 para AWS y Azure y NFSv3 para Google Cloud. NFSv4 tiene un mejor rendimiento que la versión anterior, aunque en la práctica no hay una diferencia significativa ya que ambos protocolos van a poder soportar el almacenamiento de la carga que se suministre.
- Almacenamiento en frío. Todas ellas ofrecen un servicio para el almacenamiento de datos que raramente se van a volver a utilizar, pero por ello no dejan de ser relevantes. El almacenamiento en frío suele utilizar para el almacenamiento de copias de seguridad de las bases de datos y de ciertos ficheros.

Entre los tipos de almacenamiento que se han definido en este apartado, cabe destacar que las tres plataformas disponen de todos los servicios de almacenamiento con pequeñas variaciones.

Características de seguridad

La seguridad es una de las características más importantes que debe tener una plataforma de computación en la nube. Si no disponemos de un buen sistema de seguridad, los datos almacenados serán vulnerables a los posibles ataques sobre la nube.

Una de las medidas adoptadas por todas las plataformas es la codificación de los datos, solo pudiendo obtener los datos en texto plano a través de claves de desencriptación. Las tres plataformas presentan el mismo tipo de cifrado AES 256 bits, uno de los cifrados más potentes en la actualidad.

Otra medida para proteger la infraestructura es la creación de cortafuegos para evitar intrusiones en la misma. Las tres plataformas presentan servicios de cortafuegos que tienen la misma funcionalidad, aunque cada una de ellas los citan de diferente manera.

3 Descripción del sistema

Este capítulo se centra en la descripción general del sistema de predicción. Se van a exponer detalladamente los diferentes módulos involucrados en el sistema que podemos resumir en el módulo de ingesta de datos en tiempo real e identificación y el módulo de personalización. Hay que destacar que el sistema se ha desarrollado en torno a la plataforma *cloud*, denominada Google Cloud Platform, ya que es la plataforma que mejor estructurados tenía los datos gracias a Google Analytics.

El proyecto realizado parte del módulo de personalización, ya que el módulo de ingesta de datos ya había sido implementado por la empresa LUCE. Este módulo de ingesta de datos en tiempo real se va a explicar para dar conocimiento de las variables que se han utilizado para realizar el sistema. En primer lugar, se hará una descripción breve sobre de donde provienen los datos de los clientes, a continuación, se analizarán las variables para ver cuáles son las posibles entradas con las que trabajará el modelo de predicción, y por último, se expondrá un gráfico donde se mostrarán las variables utilizadas conforme al *feature importance*.

A partir de esos datos recogidos se empieza a desarrollar el proyecto. El objetivo es la creación de un entorno que simule predicciones de probabilidades de compra en tiempo real. Para ello, se ha diseñado una arquitectura que cuenta con una API flask donde subiremos los datos de los clientes que posteriormente se lanzarán contra el modelo de predicción.

En las siguientes secciones se exponen ambos módulos, en primer lugar, el módulo de ingesta de datos donde se realiza la descripción de los datos que recogió LUCE con la plataforma de Google Analytics y el módulo de personalización que fue el punto de partida para el desarrollo del presente proyecto.

3.1 Módulo de ingesta de datos en tiempo real e identificación

Este paquete fue implementado por la empresa LUCE para llevar a cabo el diseño y el desarrollo de una arquitectura que permita la ingesta y el tratamiento de los datos en tiempo real. Se tuvieron en cuenta los siguientes aspectos: tener un histórico del cliente para realizar las predicciones, obtener las sesiones de los clientes desplegadas por las acciones realizadas en la página web, y contar con un identificador único de cliente.

3.1.1 Origen de los datos

La recopilación de datos en tiempo real la realizaron a partir de la herramienta Google Analytics, plataforma online desarrollada por Google para medir y analizar los movimientos que surgen en un sitio web o en una aplicación. El funcionamiento de esta herramienta se basa en diferentes etiquetas JavaScript y cookies, además del navegador que utilice el usuario ya que es donde se guardan las cookies.

Los datos procedentes de Google Analytics son datos de sesiones de diferentes clientes que han accedido a los servicios de la empresa, en este caso una empresa cliente a través de su página web. Algunos de los datos recogidos en Google Analytics se presentan a continuación.

- Datos de las sesiones del cliente, páginas que ha visitado, tiempo que permanece en cada una de las páginas visitadas, número de páginas que ha visitado, …

- Dispositivo desde donde ha realizado sus consultas, navegador que ha utilizado, idioma con el que ha accedido a la página y todos los parámetros que involucran a los aspectos no funcionales.
- Geolocalización desde donde ha realizado las actividades en la página web, como el continente, país, región, municipio, longitud, latitud, …
- Todos los datos relacionados con las acciones que los clientes realizan en las determinadas páginas, como *clicks* realizados por página, productos visitados, categoría de los diferentes productos, e incluso el proceso guiado de las compras que se realizan.
- Transacciones realizadas en el comercio electrónico tomando todas las compras realizadas por el usuario, el tipo de producto comprado, importes, números de pedidos, y demás aspectos relacionados con las transacciones que haya podido realizar.

3.1.2 Variables consideradas

Estas variables procedentes de Google Analytic son variables independientes (*features*) y son las variables que va a recibir el modelo de predicción. La variable dependiente (*target* o variable objetivo) es la variable a predecir. Para el caso en el que se va a trabajar la variable dependiente es un parámetro que nos muestra la probabilidad de compra de un determinado cliente para un producto de una categoría concreta. Y, a partir de esa probabilidad de compra se podrá predecir si el cliente es posible comprador o no.

Los diferentes usuarios de la página web pueden tener sesiones con antigüedades diferentes, pueden ser de hace un día, una semana, un mes o incluso de un año entero. Todas las sesiones sirven como variables de entrada al modelo, sin embargo, los parámetros de las sesiones, como el tipo de producto visitado, páginas vistas o aspectos de interés para el usuario pueden variar con el tiempo. Por ello, las *features* se han calculado solo utilizando sesiones con un período determinado de días anteriores.

Para determinar el período de tiempo LUCE efectuó diferentes pruebas con 30,15 y 7 días. Finalmente eligió 15 días anteriores que será el histórico de los usuarios que se utilizará en el modelo para realizar la predicción junto con los datos de la sesión en tiempo real.

3.1.3 Variables calculadas

Las variables calculadas son aquellos parámetros que se consideraron que pudiesen ser las entradas al modelo. Las variables que se propusieron como entradas del modelo han sido:

- *num_sesiones*: número de sesiones diferentes donde se visitan productos de diferentes categorías.
- *avg_cat_sessions*: número de sesiones diferentes con páginas visitadas de una categoría de producto determinada.
- *num_pageviews:* número de páginas vistadas relacionadas con una categoría de producto determinada.
- *avg_pageview_per_session:* promedio de páginas visitadas de una categoría de producto determinada por cada sesión.
- *dias_transcurridos*: días transcurridos desde la última página visitada a una categoría de producto determinada.

- *avg_page_time:* promedio de tiempo de páginas visitadas de una categoría de producto determinada en una sesión.
- *num_elem_carrito*: número de elementos que se han añadido al carrito, pero no comprados de una categoría de producto determinada.
- *num_elem_comprados*: número de elementos comprados de una categoría de producto determinada.
- *compra_anterior*: variable booleana que indica si el cliente ha comprado anteriormente un producto de la categoría determinada.
- *num_sesiones_totales*: número de sesiones totales independientemente de las categorías de producto visitadas.

Además, se propusieron las variables objetivo:

- *prediction*: predicción de compra de un producto de una categoría determinada. Tomará el valor 1 si el cliente es potencial comprador y 0 en caso contrario.
- *predicted_prob*: predicción de la probabilidad de compra de un producto de una categoría determinada. Tomará valores entre 0 y 1.
- *predicted_no_prob:* predicción de la probabilidad de no compra de un producto de una categoría determinada. Es el opuesto a la predicción de la probabilidad de compra.

3.1.4 Variables utilizadas en el modelo

Dentro de todas las variables calculadas se descartaron algunas de ellas por no tener relevancia en la predicción.

Las variables *avg_pageviews_per_session* y *avg_cat_sessions* se eliminaron como entradas del modelo ya que estaban directamente relacionadas con *num_sesiones* y *num_pageviews*. Además, condicionaban el modelo ya que los usuarios con pocas sesiones tenían mejores resultados, perjudicando los usuarios que contaban con muchas sesiones.

La variable *num_sesiones_totales* no tenía relevancia en el modelo ya que era una variable independiente de la categoría del producto y solo generaba ruido.

La última variable que se ha descartado ha sido *compra_anterior* ya que estaba directamente relacionada con *num_elem_comprados*, ya que la primera de ellas solo indicaba si había compra o no y la segunda muestra el número de productos comprados.

Como consecuencia de los descartes de las variables mencionadas anteriormente, las variables elegidas para que sean las entradas del modelo han sido: *num_sessiones, num_pageviews, días_transcurridos, avg_page_time, num_elem_carrito* y *num_elem_comprados.*

Para obtener la correlación de las variables calculadas con las variables objetivo LUCE utilizó un método denominado *feature importance*. Con este método se mide la importancia de una característica a partir del error de predicción de modelo después de variar esa característica. Es decir, una característica tendrá importancia en el modelo si varía el error significativamente al permutarla. Para ello, se ha aplicado el algoritmo de *random forest* y se ha analizado el parámetro de *Gini importance*. En la Figura 16 se muestra el gráfico de *feature importance* de las variables elegidas y la correlación que presentan con las variables objetivo, siendo la variable *num_pageviews* la variable con mayor importancia.

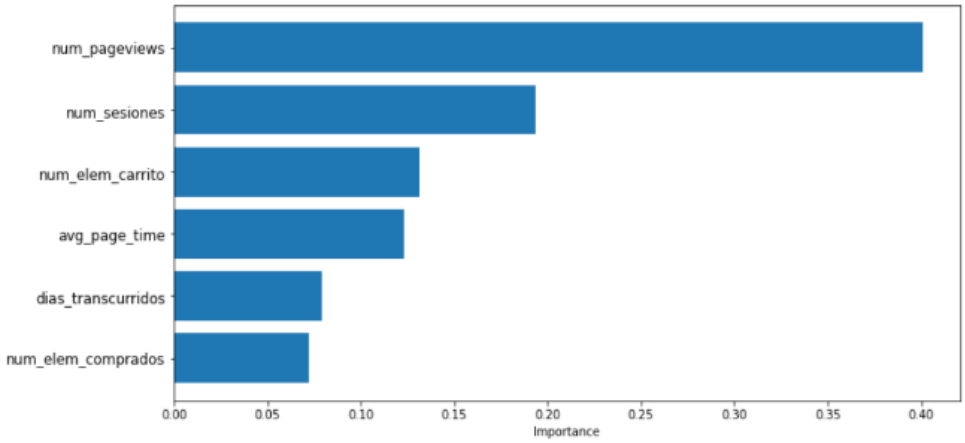


Figura 16- Feature importance de las variables del modelo.

En este paquete se ha expuesto cómo y de dónde provienen los datos que se analizan. Además, se han analizado las variables posibles como entradas en el modelo de predicción, así como el estudio de las variables que más correlación tienen con las variables objetivo y por tanto han sido las elegidas. En el siguiente módulo, el de personalización, analizaremos en detalle el modelo de predicción, así como el algoritmo utilizado para llevarlo a cabo.

3.2 Módulo de personalización

El módulo de personalización se ha creado para llevar a cabo el diseño y desarrollo de la arquitectura del sistema predictivo.

El esquema planteado para realizar el sistema de simulación de tiempo real se muestra en la Figura 17. En este diagrama de flujo se visualizan varios módulos. En primer lugar, se debe recoger una a una las sesiones de los clientes cuyos datos se encuentran en Google Analytics. Las lecturas se harán en una máquina local realizando una conexión con la plataforma de Google Cloud, a través de BigQuery. Una vez que se haya recopilado la información pertinente en la máquina local se debe realizar una petición POST a una API flask para simular que la predicción se realiza en tiempo real. El último módulo desarrollado es el modelo de predicción, el cual recogerá los datos de la sesión proveniente de la API y hará una evaluación de la predicción de compra del cliente en base al histórico de Google Analytics.

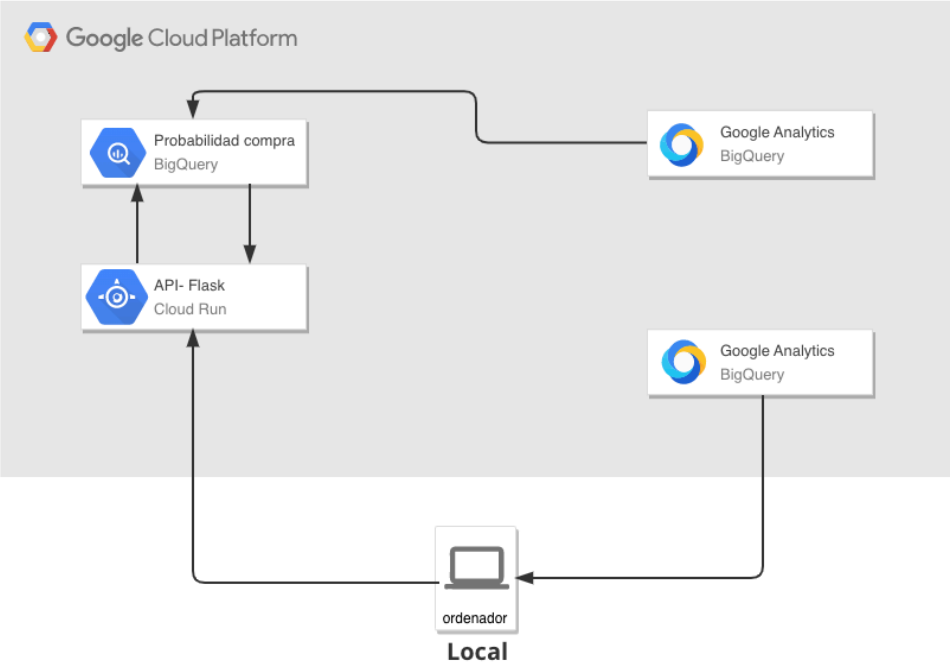


Figura 17- Diagrama de flujo que representa el sistema de tiempo real del sistema predictivo.

En los apartados que siguen se mostrarán los pasos que se han seguido para la realización de cada una de las partes del diagrama mostrado anteriormente.

3.2.1 Lectura de información de Google Analytics

Los datos de las sesiones de los clientes se han obtenido a partir de Google Analytics como se explicó en la sección 3.1.1. Se han utilizado las muestras de datos recogidas a lo largo de 2 semanas y almacenadas en una tabla de BigQuery, llegando a un total de 4.702.545 sesiones de clientes. El siguiente paso por realizar es la lectura de cada una de las sesiones que se han almacenado en Google Cloud. Para ello se ha creado un script de Python con las sentencias necesarias que explicaremos a continuación. Previamente se han instalado diferentes paquetes en la máquina local para poder realizar estas conexiones con Google Cloud.

En primer lugar, se creó un entorno virtual que permite que los paquetes de Python se instalen en una ubicación aislada, así quedan separadas las dependencias de diferentes proyectos, ya que las recomendaciones exponen que se use un entorno virtual por cada proyecto que se realice. Los pasos que seguir para la creación de un entorno virtual son:

1. Utilización del comando `venv` para la creación de una copia del paquete de instalación de Python, que se guardará en una carpeta en el directorio donde se encuentre el proyecto. Los comandos necesarios para su creación se muestran en la siguiente sentencia.

```
cd directorio_proyecto
python -m venv directorio_nuevo
```

2. Una vez creado el entorno virtual, procedemos a acceder a él para hacer las instalaciones necesarias en su interior. La activación del entorno virtual se puede ver en la siguiente sentencia de comandos.

```
source directorio_nuevo/bin/actívate
```

3. Ahora estaremos en disposición de instalar los paquetes necesarios sin afectar a los demás proyectos con los que se estén trabajando. Por tanto, se deben instalar los paquetes que se requieren para realizar las conexiones desde la máquina local a la plataforma de Google Cloud mediante las siguientes sentencias de comandos.

```
pip3 install --upgrade google-cloud-bigquery
pip3 install google-api-python-client
```

   La primera sentencia instala el paquete de BigQuery, en donde se encuentran las bibliotecas de Cloud para Python y la segunda instala el paquete de cliente, ambos de Google Cloud.
4. Una vez que se haya terminado de realizar las instalaciones en el entorno virtual y se quiera desactivar se debe realizar a través del comando `deactivate`.

En segundo lugar, se necesitan las credenciales del proyecto de BigQuery para poder realizar la conexión segura desde la máquina local y la plataforma. Estas claves se encuentran en Google Cloud en un fichero *json* que se debe descargar y guardar en el mismo directorio en el que se encuentran los scripts del proyecto.

Una vez realizados los pasos anteriores ya se estará preparado para poder realizar el script de la lectura de los datos. En la Figura 18 se muestra el código generado para hacer la petición a la plataforma de los datos de una sesión que comentaremos a continuación.

En primer lugar, se debe establecer la variable de entorno *GOOGLE_APPLICATION_CREDENTIALS* que serán las claves para realizar la conexión con la plataforma, por ello se iguala al fichero *json* descargado que hemos guardado en el directorio del proyecto. En segundo lugar, creamos un cliente de BigQuery que es el que realizará la petición a través de la función *bigquery.Client()*.La lectura de los datos se efectúa a través de una petición SQL donde se obtendrán los datos más significativos de la sesión con los que se ejecutará la predicción. En concreto, los datos que más interesan son el identificador de la visita (*visitId*), el día al que corresponde la sesión (*date*), el identificador del cliente (*clientId*) y las acciones que realiza el cliente durante la sesión (*hits*). La consulta se ha programado para que se lleve a cabo dentro de un bucle infinito para simular que todo el tiempo llegan datos al modelo de predicción. Además, cada vez que se ejecuta la consulta recogerá diferentes datos de sesiones ya que se realiza de forma aleatoria.

```python
import os
import requests
import json
from google.cloud import bigquery

# credenciales de google cloud
os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = "bq-luce-bd4bd5286194.json"
# cliente de bigQeury
client = bigquery.Client()
while (1):
    # query para obtención de una fila de la tabla de datos: API request
    query_job = client.query(
        """SELECT
      t.visitorId             AS visitorId,
      t.visitNumber           AS visitNumber,
      t.visitId               AS visitId,
      t.visitStartTime        AS visitStartTime,
      t.date                  AS date,
      t.totals                AS totals,
      t.hits                  AS hits,
      t.fullVisitorId         AS fullVisitorId,
      t.userId                AS  userId,
      t.clientId              AS clientId,
      t.channelGrouping       AS channelGrouping,
      t.socialEngagementType  AS  socialEngagementType,
    FROM `bq-luce.archiever.ga_sessions` AS t
    ORDER BY RAND() LIMIT 1""")

    # resultado de la query que se ha realizado a BigQuery
    results = query_job.result()  # Waits for query to finish

    # Conversión de los datos de una fila a formato JSON
    records = [dict(row) for row in query_job]
    result_json = json.dumps(str(records))
```

Figura 18- Script de python utilizado para hacer la consulta de la lectura de datos de una sesión.

3.2.2 Implementación de la API flask

Los datos recogidos en la consulta realizada se deben enviar a un servicio web, por ello se ha implementado una API flask. Flask es un marco web de Python que se ajusta a las necesidades que buscamos, puesto que proporciona las herramientas y funciones útiles para crear entornos web con facilidad. Solo con un simple fichero de Python podemos crear una aplicación web de forma efectiva y rápida.

Los pasos por seguir para llevar a cabo la implementación de este servicio son los siguientes.

1. Instalación de flask en el entorno virtual creado en el apartado anterior con el paquete pip.

   ```
   pip install flask
   ```

   Para verificar que la instalación se ha efectuado correctamente se puede utilizar el siguiente comando.

   ```
   python -c "import flask; print(flask.__version__)"
   ```

2. Para el lanzamiento del servicio se deben configurar algunas variables de entorno. Primero se indicará a flask donde encontrar el fichero de código del servicio con la siguiente sentencia.

   ```
   export FLASK_APP= nombre_fichero_API
   ```

A continuación, se indicará que el tipo de ejecución debe ser de desarrollo.

```
export FLASK_ENV=development
```

Por último, se puede ejecutar el servicio a través del siguiente comando.

```
flask run
```

3. Como último paso se debe verificar si el servicio se ha activado correctamente. En la Figura 19 se muestra la salida por consola de comandos como consecuencia del lanzamiento del servicio.

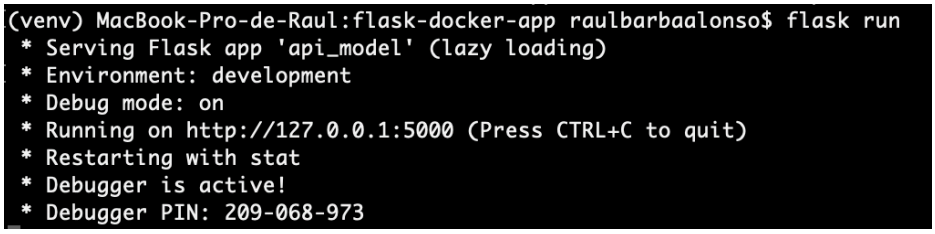


Figura 19- Resultado del lanzamiento del servicio en consola de comandos.

El resultado obtenido presenta la siguiente información que puede ser útil al usuario.

- El nombre del servicio y el entorno en el que se está ejecutando.
- *Debug mode: on* nos muestra que el depurador de flask está activado y puede dar realimentación de los errores que proporciona el servicio si hay problemas.
- El servicio está corriendo en la URL http://127.0.0.1:5000, es decir la dirección IP que corresponde con localhost en el puerto 5000.

Dado que ya se ha expuesto como instalar y lanzar el servicio flask, ahora se va a mostrar cómo se ha estructurado internamente y cómo se efectúa la subida de los datos leídos al servicio web. La estructura del servicio web presenta dos recursos, el recurso raíz y el recurso datos donde enviaremos los datos de la lectura. En la Figura 20 se puede ver la definición del servicio junto con los dos recursos. Se puede apreciar que el recurso datos acepta dos métodos (GET y POST), esto es porque los datos se van a enviar por peticiones http y debe aceptar el método POST además del GET.

```python
# Definición de la aplicación
app = Flask(__name__)


# Recurso raiz
@app.route('/')
def raiz():
    return {"Directorio": "raiz"}


# recurso datos a donde subimos los datos de la BigQuery
@app.route("/datos", methods=['GET', 'POST'])
def post():
    content: object = request.json
```

Figura 20- Código de la estructura de servicio flask creado.

La subida de los datos se realiza a partir de una petición http utilizando el método POST. El código desarrollado se muestra en la Figura 21, donde podemos ver que se utiliza la función *request.post()* de la librería *requests* que se deberá importar al comienzo del fichero. Dicha función presenta tres atributos: la uri a donde se desea subir los datos, los datos propiamente dichos que tienen formato *json* y las cabeceras de la petición, las cuales deben de ser compatibles con el formato de datos que se envíen.

```python
# Uri base del servidor Flask
baseUrl = 'http://localhost:5000'
#baseUrl='https://app-achiever-dey6vjv7bq-ey.a.run.app'
# Cabecera de json para realizar la petición
headers = {'Content-Type': 'application/json; charset=utf-8'}
# petición POST para el envío de datos al servidor
r = requests.post(baseUrl + "/datos", data=result_json,headers=headers)
```

Figura 21- Código de petición http POST para la subida de los datos al servicio web.

Ahora que se ha desarrollado el servicio web debemos desplegarlo como una instancia Docker. Docker es una plataforma de software la cual permite la implementación, creación y realización de pruebas de aplicaciones y servicios en un modo rápido y efectivo. Por lo que la lista de ventajas de porqué utilizar Docker puede llegar a ser muy larga, pero se pueden resumir las más importantes en las que siguen:

- Permite tener una respuesta con mayor rapidez.
- Reduce costes de recursos ya que Docker suministra solo los recursos específicos que requiere una aplicación o servicio, y conlleva ahorrar ya sea en mantenimiento o en sobrecarga de servidores.
- El servicio web se ejecuta dentro de un contenedor y, por tanto, el contenedor se puede ejecutar en cualquier sistema operativo. No se tiene que configurar y programar el software o sistemas operativos que sean compatibles para que se pueda llevar a cabo su ejecución.

Para llevar a cabo el despliegue de Docker se han seguido los siguientes pasos.

1. Creación de un *Dockerfile*, fichero de texto en el que se indica las instrucciones que debe de seguir para hacer el ensamblado y generar la imagen de Docker. Las directrices más relevantes del fichero son la indicación de la ruta del directorio

donde se encuentra el proyecto, instalar las dependencias dentro de la imagen que se obtienen del fichero requierements.txt y lo más importante informar a Docker de los comandos que tiene que ejecutar cuando la imagen se ejecute dentro del contenedor. En nuestro caso se le indicará que ejecute `flask run` para iniciar el servicio flask en la ruta y puerto correspondiente.

2. Obtención del fichero requierements.txt a través del siguiente comando.

```
pip3 freeze > requirements.txt
```

3. Instalación de las dependencias en el fichero requerements.txt que es el que copiaremos en la imagen a través del *Dockerfile*.

```
pip3 install --file requierements.txt
```

4. Creación de la imagen Docker usando el siguiente comando.

```
docker build --tag python-docker
```

5. Una vez generada la imagen Docker se pueden listar todas las imágenes que se han creado para verificar todo ha ido correctamente a través del siguiente comando.

```
docker images
```

6. Para realizar la ejecución de Docker se utiliza el siguiente comando (-dti para que se ejecute en segundo plano, si no se tendría que abrir una segunda terminal).

```
docker run -dti --publish 5000:5000 python-docker-api
```

Aprovechando que contamos con las funcionalidades y servicios ofertados por la plataforma de Google Cloud se expuso la posibilidad de poder integrar el contenedor Docker en una Plataforma como Servicio (PaaS). Como se expuso en el apartado 2.3.1 es un entorno de desarrollo e implementación de aplicaciones, y puede albergar servidores, redes y almacenamiento. Las ventajas por las que utilizar este tipo de plataformas se exponen a continuación.

- Los costes se reducen en todo el proceso de creación de la aplicación, con una suscripción de pago por uso se dispone de un software refinado entre otras funcionalidades para poder trabajar.
- La información se comparte en múltiples zonas geográficas, por lo que se puede acceder simultáneamente a los servicios.

- Mantenimiento por parte del proveedor de la plataforma con lo que se reducen los gastos que pueda suponer.
- Administración de las funcionalidades de la aplicación desarrollada durante el ciclo de vida en la plataforma como pueden ser compilación, implementación, ejecución, pruebas y actualización.

El proyecto que se está desarrollando está soportado en gran medida por microservicios de Google, por tanto, se va a realizar una comparativa de las dos alternativas de plataformas en la nube que ofrece Google para alojar nuestra API: Cloud Run y App Engine.

App Engine es una plataforma sin servidores que surgió como alternativa para el alojamiento de aplicaciones web. Además del alojamiento, permite utilizar varios lenguajes, bibliotecas y *frameworks*. La plataforma es la encargada del abastecimiento de servidores sin tener que involucrar al usuario. Al igual que App Engine, Cloud Run también es una plataforma sin servidores, encargada especialmente de la compilación de aplicaciones.

En cuanto a las similitudes de ambas plataformas cabe destacar. Ambas son compatibles con cantidad de lenguajes de programación como Python, Java, Node.js, etc, permiten peticiones HTTP para realizar las comunicaciones cliente-servidor, son plataformas destinadas para el alojamiento y ejecución de aplicaciones sin servidor y ambas presentan un coste de pago por uso. Dentro de las diferencias, Cloud Run puede implementar contenedores en Google Kubernetes Engine (GKE) encargado de la implementación y administración de las aplicaciones obteniendo mejores prestaciones, como menor sobrecarga (no incluyen imágenes del sistema operativo), funcionamiento más constante y mayor eficiencia.

La toma de decisión se debe hacer en función de si se quiere que la aplicación se ejecute dentro de un contenedor o si se considera que no es necesario la ejecución dentro del contenedor. Si se requiere la ejecución dentro de un contenedor, ya sea Docker Kubernetes, se debe utilizar Cloud Run, pero por el contrario si no se necesita la implementación de contenedores se debe elegir App Engine. Por tanto, la elección ha sido utilizar Cloud Run como solución PaaS ya que se van a utilizar contenedores para la implementación.

Una vez elegida la alternativa de Cloud Run se debe subir la imagen de la instancia creada a la plataforma y ver su correcto funcionamiento. Para ello se ha hecho uso de la consola de Google Cloud (*Cloud Shell*). Los comandos utilizados para llevarlo a cabo han sido los siguientes.

1. Verificar que nos encontramos en el directorio del proyecto correcto dentro de la plataforma.

   ```
   $ gcloud config set project nombre_proyecto
   ```

2. Comprimir el directorio donde tengamos el proyecto en un zip para poder subirlo a la plataforma. Una vez subido se debe descomprimir desde *Cloud Shell* a través del comando `unzip name.zip` y se navega hasta el directorio donde se encuentra el *Dockerfile*.
3. Compilar la imagen a través del siguiente comando.

```
$ gcloud builds submit --tag eu.gcr.io/nombre_proyecto/app_name --project nombre_proyecto
```

4. Ejecutar la imagen con diferentes atributos como se muestra en la siguiente sentencia.

```
$ gcloud run deploy app_name --image eu.gcr.io/nombre_proyecto/app_name --platform managed --region europe-west3 --project nombre_proyecto --allow-unauthenticated --cpu 1000m --memory 512Mi --timeout 60m --service-account="cuenta_de_servicio"
```

   En esta sentencia se pueden ver algunos parámetros relevantes como los usuarios que están autentificados para usar la aplicación, le velocidad de procesamiento, el almacenamiento o el tiempo límite de respuesta.
5. Por último, falta vincular el servicio a IAM, que es el que realiza el control e identificación de acceso para administrar los recursos. Esto se lleva a cabo a través de la sentencia siguiente.

```
$ gcloud beta run services add-iam-policy --region=europe-west3 --member=allUsers --role=roles/run.invoker --platform=managed --project nombre_proyecto app_name
```

Una vez que se han llevado a cabo todos estos procesos se podrá ver el funcionamiento de la aplicación en Cloud Run dentro de la plataforma de Google Cloud. Allí encontraremos el servicio creado con la uri que se le ha asignado a nuestra aplicación. Hay que destacar que se debe realizar un cambio en el script de Python en el que realizamos la subida de los datos a la API, donde la uri ya no será localhost sino que se modificará por la uri que se ha generado en el servicio de Cloud Run.

3.2.3 Implementación del modelo predictivo

En esta sección se va a exponer la discusión del tipo de algoritmo que se ha utilizado en el modelo de predicción global contando con las sesiones globales del cliente. El objetivo de este modelo es dotar de una probabilidad de compra a un usuario concreto durante una de sus sesiones contando con sus históricos. Aunque el problema también se puede entender como una clasificación binaria, en la que el modelo predice si el cliente va a comprar o no.

Dado que la problemática se puede ver de dos puntos diferentes, se ha considerado que se va a utilizar un algoritmo de clasificación ya que este tipo de algoritmos puede evaluar tanto casos binarios como casos que no lo son.

Además, se debe tener en cuenta que se parte de datos etiquetados, como son los históricos de los clientes, y con ellos se puede entrenar el modelo. Por ello, se trata de un problema de aprendizaje supervisado que se resolverá con un algoritmo de clasificación en la que la variable que interesa es la probabilidad de que un usuario compre un producto de una categoría determinada.

Identificado el problema a resolver, se deben plasmar los algoritmos que mejor se ajusten para el desarrollo del modelo. A continuación, mostraremos la variedad de algoritmos que se han evaluado y la elección del que se ha utilizado.

Hay que destacar que la elección del algoritmo del modelo de predicción es anterior a la realización del proyecto. Esto se debe a que la empresa con la que se ha colaborado ya contaba con diferentes proyectos que utilizaban y evaluaban los algoritmos. Por tanto, esta sección pone en contexto como ha sido la elección del algoritmo, pero no ha sido realizada dentro del trabajo expuesto.

A partir del sistema de predicción global desarrollado por LUCE, se caracterizará con las variables de entrada necesarias para que el modelo de predicción sea personalizado para cada cliente.

3.2.3.1 Algoritmos descartados

Se han descartado dos algoritmos debido a que no se espera tener mejores resultados que con otros, además de su simpleza. Estos algoritmos son el algoritmo de *Naive Bayes* y los árboles de decisión (*decision trees*).

El algoritmo de *Naive Bayes* es ingenuo como su propio nombre indica ya que considera las variables independientes entre ellas, por lo que se entiende como un mal estimador para el caso que nos ocupa.

Los árboles de decisión tienen la particularidad de que tienden al *overfitting*, que como se explicó en el apartado 2.2.1.5, es el problema debido al excesivo entrenamiento del modelo con datos similares y en el caso de que los datos disten lo más mínimo del conjunto de datos de entrenamiento el error es muy significativo. Por ello, se suelen utilizar métodos para contrarrestar este efecto como el *bagging* que utiliza el algoritmo de *random forest*.

3.2.3.2 Algoritmos evaluados

Logistic Regresion

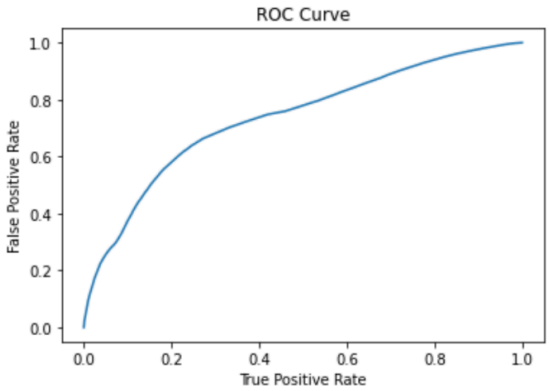
Es un algoritmo que utiliza una técnica bastante sencilla, y puede ser por ello que se utilice con frecuencia. Una de las principales ventajas que presenta es que los resultados que se obtienen son altamente interpretables, a diferencia de muchos otros que se consideran como "cajas negras". Como desventaja hay que destacar que no es posible resolver directamente problemas no lineales, es decir, para que el algoritmo sea efectivo los datos deben ser linealmente separables.

Se han realizado diferentes pruebas modificando algunos de los parámetros del algoritmo. En concreto se ha variado el parámetro *maxIter*, que indica el número de iteraciones máximas que se permiten durante el tiempo que dura el entrenamiento del modelo, *regParam* y *elasticNetParam.* En concreto, en la Figura 22 se pueden ver las métricas que se han obtenido para un *maxIter*=100, un *regParam*=0,001 y un *elacticNetParam*=0,01.

	precision	recall	f1-score
0.0	0.71	0.85	0.78
1.0	0.69	0.49	0.57
accuracy			0.71
macro avg	0.70	0.67	0.67
weighted avg	0.70	0.71	0.69

	precision	recall	f1-score
0.0	0.71	0.85	0.78
1.0	0.69	0.49	0.57
accuracy			0.71
macro avg	0.70	0.67	0.67
weighted avg	0.70	0.71	0.69

Accuracy: 0.7071661396781664



Training Area Under ROC: 0.736560274480623

Test Area Under ROC 0.7365596382407402

Figura 22- Métricas obtenidas para el algoritmo de regresión logística.

En esta figura se puede ver a la izquierda de la curva ROC dos matrices, la de arriba representa las métricas obtenidas con el conjunto de datos de entrenamiento y la de abajo a las métricas obtenidas en el test. En las dos matrices se visualizan las métricas precisión, *recall* y f1-score para cada una de las etiquetas. La etiqueta 0 indica que el cliente no ha realizado compra y la etiqueta 1 que sí se ha realizado compra. Además, se muestra la curva ROC mostrando el área bajo la curva de los datos de entrenamiento y del test, que como se puede observar es de 0.736, por lo que el rendimiento del modelo es bastante bueno.

Support Vector Machine (SVM)

Es una de las técnicas más utilizadas y suelen ofrecer un rendimiento más alto que la regresión logística, en mayor motivo porque maneja espacios no lineales a diferencia de la regresión logística.

La principal desventaja que nos encontramos con la aplicación de este algoritmo es la imposibilidad de estimaciones de probabilidades, sino que realiza una validación cruzada. La validación cruzada es una técnica que se basa en calcular repetidamente la media aritmética obtenida de los datos de evaluación. Este hecho es relevante ya que nuestro objetivo es predecir la probabilidad de compra de un cliente determinado.

Con la utilización de esta técnica se han obtenido mejores resultados que con la regresión descrita anteriormente. Para la realización de las pruebas se ha utilizado *LinearSVC* (*Linear Super Vector Classification*) que es un tipo de SVM. En la Figura 23 se muestran las matrices con las métricas obtenidas con el algoritmo *LinearSVC* con un máximo de iteraciones de 200, *regParam*=0,01 y *aggregationDepth*=2.

	precision	recall	f1-score
0.0	0.69	0.90	0.78
1.0	0.71	0.39	0.51
accuracy			0.69
macro avg	0.70	0.64	0.64
weighted avg	0.70	0.69	0.67

	precision	recall	f1-score
0.0	0.69	0.89	0.78
1.0	0.71	0.39	0.51
accuracy			0.69
macro avg	0.70	0.64	0.64
weighted avg	0.70	0.69	0.67

Figura 23- Métricas obtenidas para el algoritmo de LinearSVC

Random forest

Random forest es un algoritmo creado como combinación de varios árboles de decisión, y se conoce como un algoritmo ensamblado. Como ya se ha mencionado los árboles de decisión tienden al *overfitting.* Para solventar el problema se combinan los árboles de decisión y cada uno de ellos realizará una predicción, es decir, cada uno de ellos realiza un voto sobre su elección. La respuesta más votada será la respuesta del algoritmo.

El principal inconveniente de este algoritmo es que no funciona correctamente si el conjunto de datos es pequeño, pero para el proyecto realizado no va a ser un problema ya que se cuenta con un conjunto de datos bastante amplio.

Para realizar las pruebas se han utilizado los diferentes parámetros.

- *numTrees*: número de árboles de decisión con los que contará *el random forest*. Se han probado valores entre 50 y 200 árboles.
- *maxDepth*: profundidad máxima que pueden alcanzar los árboles. Se han probado valores entre 5 y 30 de profundidad.
- *minInstancesperNode*: mínimo número de instancias que alberga cada nodo en los árboles. Se han probado valores entre 20 y 50 instancias.
- *featureSubsetStrategy*: número de *features* candidatas en cada nodo.

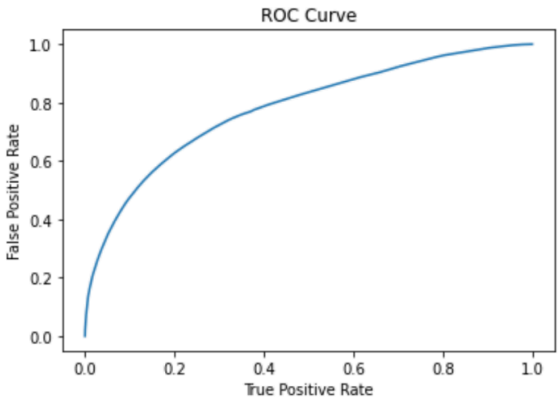
La configuración de parámetros con el que se obtiene un mejor resultado es para un *numTress*=100, *maxDepth*=20, *minInstanceperNode*=10 y *featureSubsetStrategy*=auto.

En la Figura 24 se pueden observar las métricas obtenidas con este modelo, que son superiores a las conseguidas con los modelos de regresión y SVM. En concreto, el área bajo la curva en este caso es más elevado que con los algoritmos anteriores llegando a 0.779.

	precision	recall	f1-score
0.0	0.75	0.86	0.80
1.0	0.71	0.55	0.62
accuracy			0.74
macro avg	0.73	0.70	0.71
weighted avg	0.73	0.74	0.73

	precision	recall	f1-score
0.0	0.74	0.85	0.79
1.0	0.70	0.54	0.61
accuracy			0.73
macro avg	0.72	0.69	0.70
weighted avg	0.72	0.73	0.72

Accuracy: 0.7372790024025192



Training Area Under ROC: 0.7797594295363898

Test Area Under ROC 0.7797749686170742

Figura 24- Métricas obtenidas para el algoritmo de random forest.

Para la creación y despliegue del modelo se ha utilizado BigQuery ML, ya que permite ejecutar y crear modelos de *machine learning* directamente a partir de sentencias SQL. Es decir, utilizando consultas SQL se pueden crear y ejecutar modelos de *machine learning*. Al igual que otras librerías de *machine learning*, BigQuery ML permite escoger el tipo de algoritmo como regresión logística, redes neuronales, *clustering* con k-means, árboles de regresión y clasificación, …

Algoritmo elegido

Los algoritmos expuestos en BigQuery ML se pueden clasificar como se aprecia en la Figura 25. Dependiendo la acción que se requiera realizar tenemos un tipo de algoritmo, como los que se han mencionado anteriormente. El algoritmo que mejores resultados en cuanto al rendimiento que presenta ha sido el *random forest*. Sin embargo, en BigQuery ML no se encuentra disponible *random forest*, por lo que se ha optado por una alternativa similar como es el *Boosted Tree*, el cual utiliza un algoritmo *XGBoost*.

XGBoost es un algoritmo de ensamblado al igual que *random forest.* Los algoritmos ensamblados se forman a partir de algoritmos más simples, que cuando se unen forman un algoritmo más potente. A diferencia de *random forest* que utiliza *bagging*, *XGBoost* utiliza *boosting*.

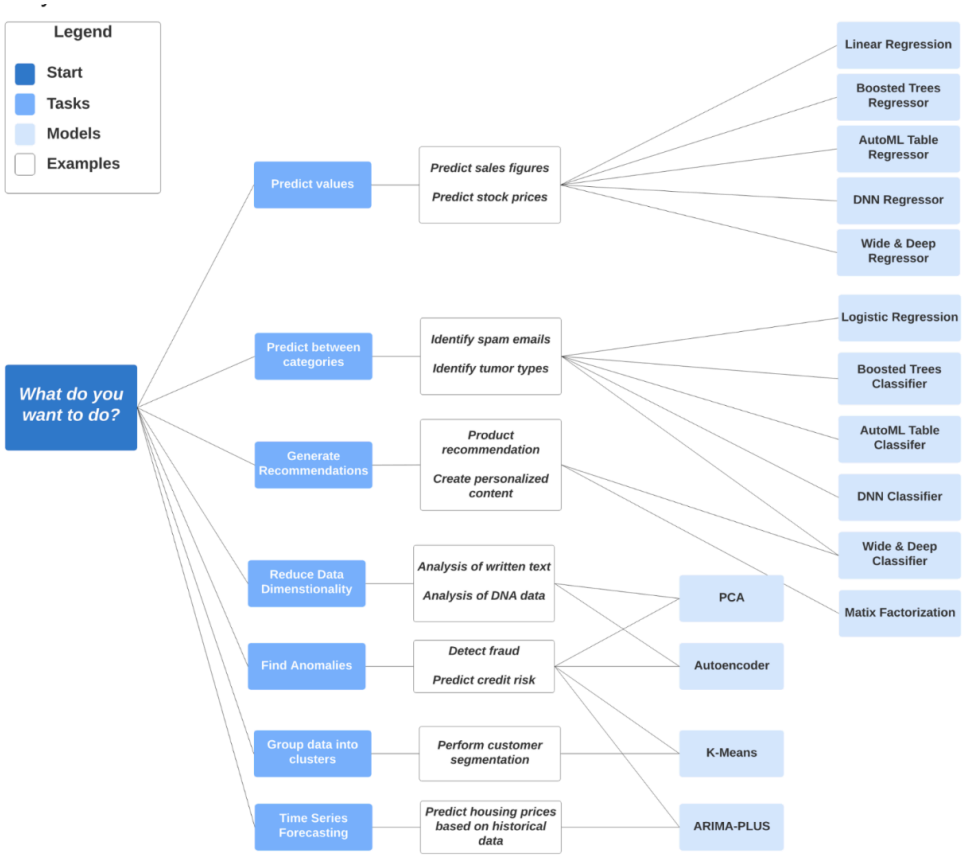


Figura 25- Esquema de los algoritmos de BigQuery ML.

El algoritmo utiliza gran cantidad de hiperparámetros para su entrenamiento. No hace falta entrar en detalle de todos ellos, pero sí es recomendable tener una visión y definición de algunos de ellos. En la Figura 26 se exponen los hiperparámetros mencionados del algoritmo *Boosted Tree*.

```
{CREATE MODEL | CREATE MODEL IF NOT EXISTS | CREATE OR REPLACE MODEL} model_name
[OPTIONS(MODEL_TYPE = { 'BOOSTED_TREE_CLASSIFIER' | 'BOOSTED_TREE_REGRESSOR' },
         BOOSTER_TYPE = {'GBTREE' | 'DART'},
         NUM_PARALLEL_TREE = int64_value,
         DART_NORMALIZE_TYPE = {'TREE' | 'FOREST'},
         TREE_METHOD = {'AUTO' | 'EXACT' | 'APPROX' | 'HIST'},
         MIN_TREE_CHILD_WEIGHT = int64_value,
         COLSAMPLE_BYTREE = float64_value,
         COLSAMPLE_BYLEVEL = float64_value,
         COLSAMPLE_BYNODE = float64_value,
         MIN_SPLIT_LOSS = float64_value,
         MAX_TREE_DEPTH = int64_value,
         SUBSAMPLE = float64_value,
         AUTO_CLASS_WEIGHTS = { TRUE | FALSE },
         CLASS_WEIGHTS = struct_array,
         L1_REG = float64_value,
         L2_REG = float64_value,
         EARLY_STOP = { TRUE | FALSE },
         LEARN_RATE = float64_value,
         INPUT_LABEL_COLS = string_array,
         MAX_ITERATIONS = int64_value,
         MIN_REL_PROGRESS = float64_value,
         DATA_SPLIT_METHOD = { 'AUTO_SPLIT' | 'RANDOM' | 'CUSTOM' | 'SEQ' | 'NO_SPLIT' },
         DATA_SPLIT_EVAL_FRACTION = float64_value,
         DATA_SPLIT_COL = string_value,
         ENABLE_GLOBAL_EXPLAIN = { TRUE | FALSE }
)];
```

Figura 26- Hiperparámetros del algoritmo Boosted Tree.

A continuación, se definen que hiperparámetros se han tenido en cuenta:

- MODEL_TYPE. Tipo de modelo a utilizar. Se ha utilizado el BOOSTED_TREE_CLASSIFIER.
- NUM_PARALLEL_TREE. Número de árboles construidos en paralelo en cada iteración. Se ha dejado el valor por defecto, que es 1, pues valores más elevados aumentaban el tiempo y coste de entrenamiento y no se conseguía mejor rendimiento.
- MAX_TREE_DEPTH. Profundidad máxima. Se han probado valores de entre 5 a 25. A partir de 10 de profundidad, el rendimiento del modelo disminuye mucho. Se ha elegido el valor de 6.
- MAX_ITERATIONS. Número máximo de iteraciones en el entrenamiento. Se ha dejado el valor por defecto que es de 20 iteraciones, ya que gracias al *early stop* el modelo finaliza su entrenamiento cuando, de una iteración a otra, no hay prácticamente mejora.
- AUTO_CLASS_WEIGHTS. Booleano para indicar si queremos balancear las clases o no. Se ha puesto igual a FALSE, ya que nosotros mismos hemos realizado el balanceo de clases mediante el *downsampling*.
- EARLY_STOP → Booleano para indicar si se quiere realizar *early stop*. Se ha puesto igual a TRUE. Cuando en el entrenamiento del modelo se termina una iteración, se calcula cuánto ha mejorado respecto a la anterior iteración. Si esa mejora es inferior al valor de MIN_REL_PROGRESS, el modelo finaliza su entrenamiento. Esto es lo que se conoce como *early stop*. El valor de MIN_REL_PROGRESS se ha dejado el que viene por defecto, que es 0,01.
- LEARN_RATE → Indica el porcentaje de cambios con el que se actualizan los pesos en cada iteración del entrenamiento. Se ha puesto igual a 0,3.
- DATA_SPLIT_METHOD → Indica el método para dividir los datos entre conjunto de entrenamiento y de validación. Se ha utilizado 'AUTO_SPLIT'. Con esta estrategia para la división, se seleccionan 10.000 filas de manera aleatoria como conjunto de evaluación, y el resto de las filas como conjunto de entrenamiento.

4 Análisis de los resultados

En este capítulo se van a analizar los resultados obtenidos tras la aplicación del modelo de predicción. Primero se hará una comparativa de los resultados que se obtuvieron con el sistema desarrollado anteriormente por la empresa LUCE en el que no se tenía en cuenta el tiempo real y las predicciones se realizaban una vez que la sesión había finalizado, con el sistema desarrollado que realiza las predicciones en tiempo real. En segundo lugar, dado que con cada *hit* (acción) que el usuario ha realizado se obtiene una probabilidad de compra diferente, se puede hacer un análisis de los resultados y ver si con cada *hit* que realiza el usuario el sistema presenta una mejor predicción.

4.1 Comparativa sistemas tiempo real/no tiempo real

En esta sección se va a realizar una comparativa del sistema de predicción desarrollado con su antecesor. Para ello se van a comparar diferentes métricas como son el *recall* y la precisión a través de diferentes curvas. Para su realización se ha realizado un barrido del umbral de probabilidad desde 0 hasta 1. Si la predicción de probabilidad de compra es menor que el umbral de decisión se clasifica como que el cliente no ha comprado, y si la predicción de probabilidad de compra está por encima del umbral se clasifica como compra.

4.1.1 Sistema sin tiempo real

En primer lugar, se van a presentar los resultados obtenidos en el sistema predictivo genérico, sin tener en cuenta tiempo real, es decir, el sistema hace una predicción en base a la sesión completa del cliente. Hay que destacar que este sistema se ha desarrollado con anterioridad al presente proyecto.

En la Figura 27 se muestra la gráfica que relaciona la precisión con el *recall* cuando se realiza un barrido del umbral. Se puede apreciar que a medida que el umbral aumenta el valor de la precisión aumenta y el valor del *recall* disminuye. Esto sucede porque si el umbral aumenta, los casos de falsos positivos decrecen y con ello el valor de la precisión crece. Sin embargo, el valor del *recall* presenta un efecto contrario, a medida que el umbral aumenta, aumentan los casos de falsos negativos, por lo que el recall decrece. En este caso se debe tener un compromiso entre el *recall* y la precisión para la elección del umbral.

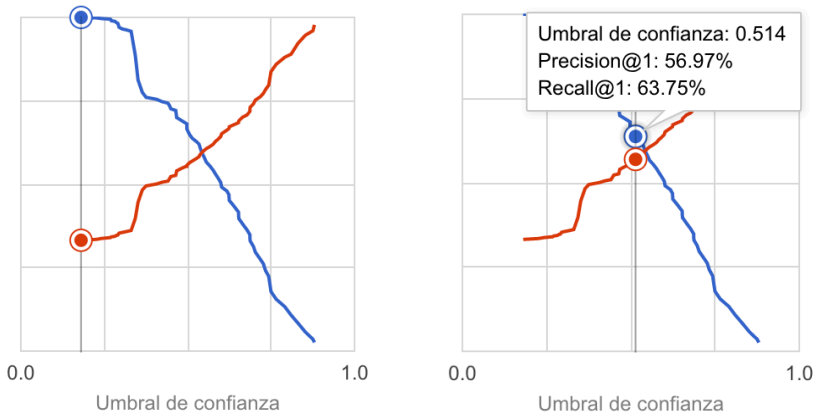


Figura 27- Gráficas de precisión-recall en función del umbral para un sistema sin tiempo real.

Además, se puede obtener la gráfica PR, que como hemos descrito en capítulos anteriores es un representación de la precisión frente al *recall*. A medida que aumenta el umbral, la precisión crece y el *recall* decrece como se aprecia en la Figura 28. En la gráfica de la izquierda tenemos un umbral de 0.514, con lo que aproximadamente la precisión y el *recall* están en equilibrio, pero si aumentamos el umbral o límite de confianza, la precisión aumenta hasta el 85%, y el *recall* se queda con valores ínfimos del 15%.

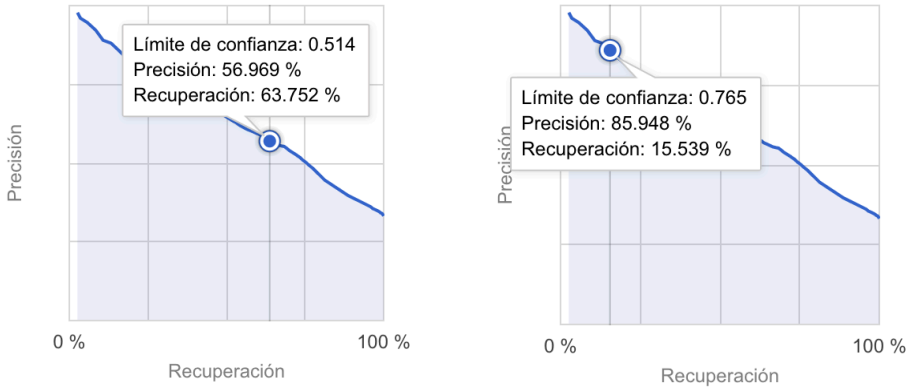


Figura 28- Gráficas de precisión-recall para el sistema sin tiempo real

Por último, se ha calculado la curva ROC, que representa la tasa de verdaderos positivos frente a la tasa de falsos positivos. La gráfica se muestra en la Figura 29, en la que se puede ver que el modelo funciona razonablemente bien, ya que el valor óptimo sería el vértice superior izquierdo, donde la tasa de falsos positivos sería creo y la tasa de verdaderos positivos sería máxima. El área bajo la curva (AUC) calculado es de 0.742 por lo que el modelo tiene una probabilidad del 74,2% de distinguir entre las clases positivas y negativas.

Figura 29- Curva ROC relacionando la tasa de verdaderos positivos frente a la tasa de falsos positivos para el sistema sin tiempo real.

4.1.2 Sistema en tiempo real

Para este sistema se van a analizar las mismas gráficas que se han mostrado para el sistema que no realiza las predicciones en tiempo real. Hay que tener en cuenta que el modelo de predicción es el mismo que el utilizado en el otro sistema, pero se particulariza

para que las predicciones no se realicen en cuanto a las sesiones sino para cada *hit* individual del cliente.

Primero evaluaremos el desempeño global del algoritmo, esto es, no distinguiremos en qué momento dentro de la sesión está interactuando el cliente. Por lo tanto tratamos todos los *hits* a la vez.

La primera de las gráficas es la que relaciona la precisión y el *recall* en función del umbral, mostrada en la Figura 30. Si hacemos una comparativa con la que se obtenía en el caso anterior, el *recall* sí que tiene una forma parecida ya que decrece en función del aumento del umbral. Sin embargo, para el caso de la precisión no aumenta si el umbral aumenta, sino que se estanca en torno a un 10%. Esto puede ser debido a que el modelo de predicción no es el apropiado para la realización de las predicciones para *hits* individuales, y tendríamos que reentrenar el modelo. También se barajan otras opciones como obtener mayores conjuntos de datos de test, ya que los valores de verdaderos positivos no es tan significativo como se pensaba en un principio, en contraposición a los falsos positivos y hace que el valor de la precisión no aumente como debería.

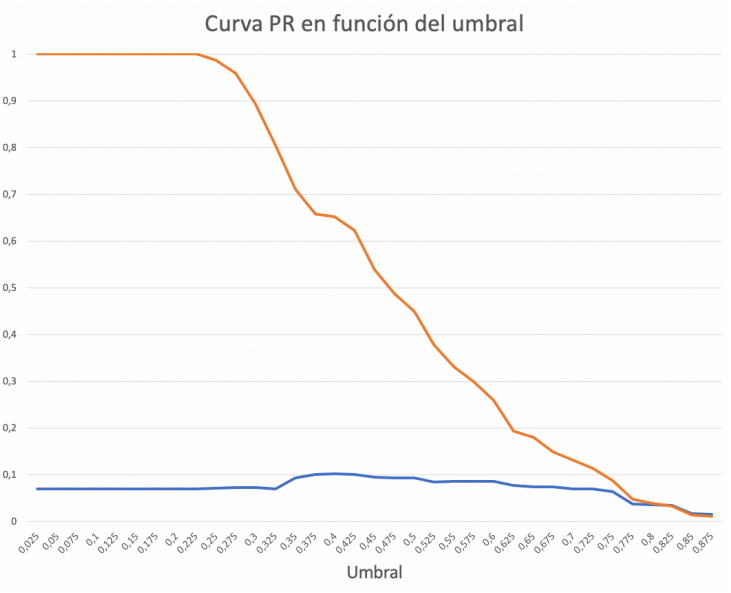


Figura 30- Gráfica precisión-recall en función del umbral para el sistema en tiempo real.

A continuación, en la Figura 31 se muestra la gráfica PR como vimos en el apartado anterior. Como se ha explicado la precisión presenta valores muy bajos entre el 2 y el 10% y con fluctuaciones, por lo que su comportamiento no es el esperado. Sin embargo, para el valor del *recall* aumenta en cuando el umbral disminuye como es lógico dado que los falsos negativos disminuyen.

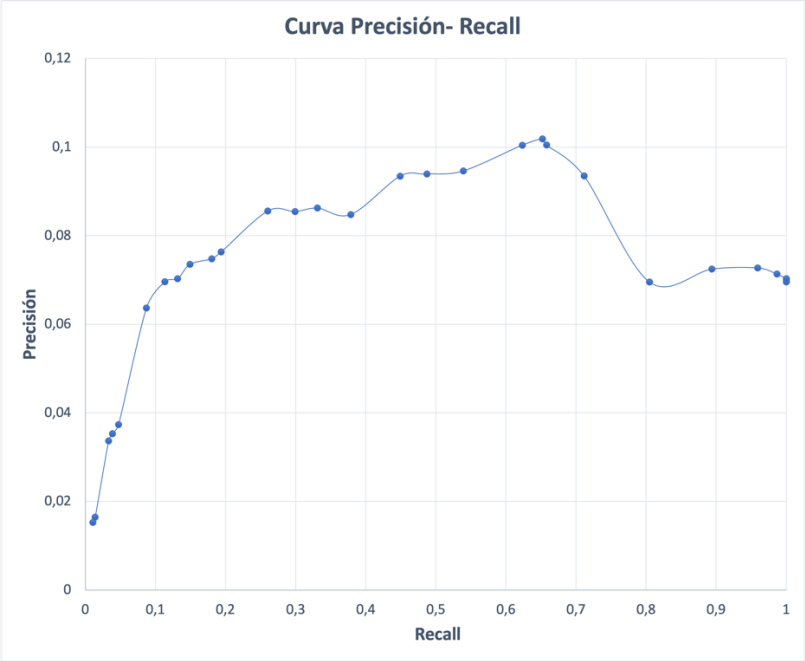


Figura 31- Gráfica de precisión-recall del sistema en tiempo real

La última de las gráficas que queda por comparar es la curva ROC mostrada en la Figura 32. Debido a que los falsos positivos también afectan a la generación de la curva ROC, no se van a obtener unos resultados tan buenos como los que se esperaba y como se tienen para el otro sistema.

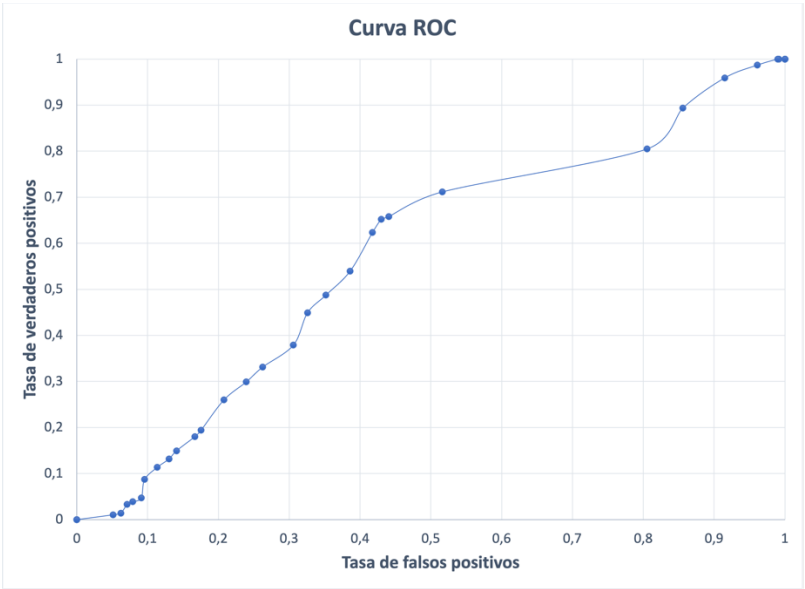


Figura 32- Curva ROC relacionando la tasa de verdaderos positivos frente a la tasa de falsos positivos para el sistema en tiempo real.

Por los motivos mencionados, la curva ROC presenta un AUC de 0.575, es decir, se tiene una probabilidad del 57,5% de distinguir entre la clase positiva y la negativa, por lo que no es un buen modelo de predicción y se tendrá que evaluar el modelo de entrenamiento para saber las causas de la mala predicción.

4.2 Análisis de los resultados por hits

Como ya indicamos en el apartado anterior, la primera evaluación del comportamiento del algoritmo se centraba en la evaluación del comportamiento global, esto es, se considera que todos los *hits* tienen la misma importancia independientemente del momento en el que se produzcan.

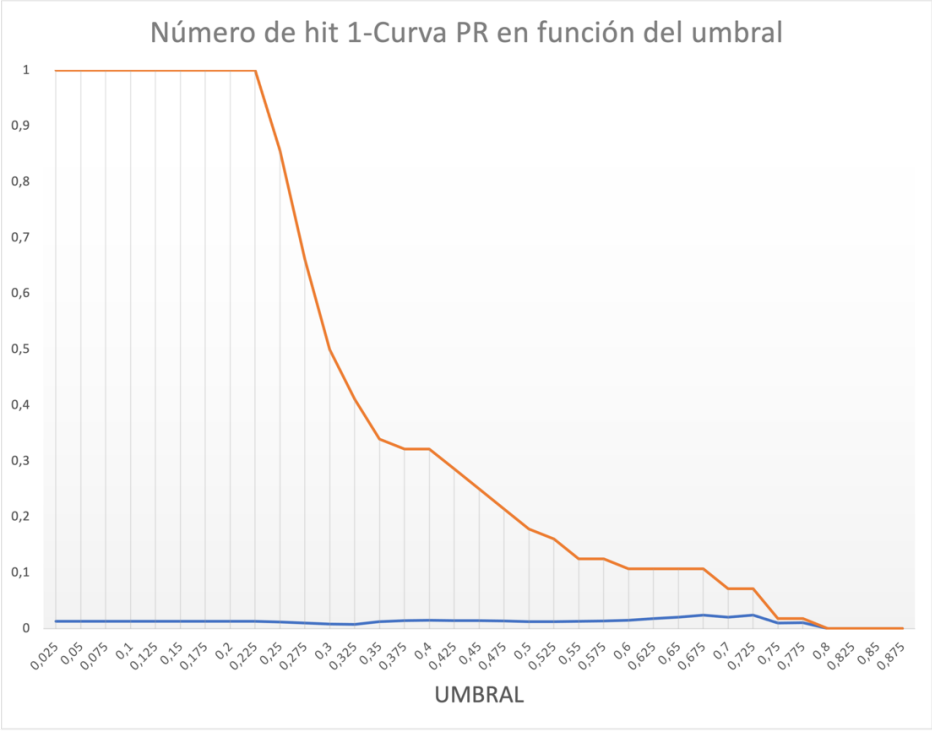
En este apartado se va a realizar un análisis de cómo es la evolución de la predicción de probabilidad de compra en función de los *hits* que realizan los clientes. Es decir, se van a evaluar las métricas y curvas que se han analizado hasta ahora, pero para el conjunto de todos los primeros *hits*, todos los segundos, todos los terceros, y así hasta un cierto número de *hits*. En concreto se ha realizado el análisis para los siguientes números de *hits*: *hits*=1, *hits*=3, *hits*=6, *hits*=10 y *hits*=20 individualmente.

La razón para llevar a cabo este análisis es que el algoritmo mantiene información sobre la sesión, por lo que según evoluciona debería ser capaz de predecir mejor las intenciones del cliente.

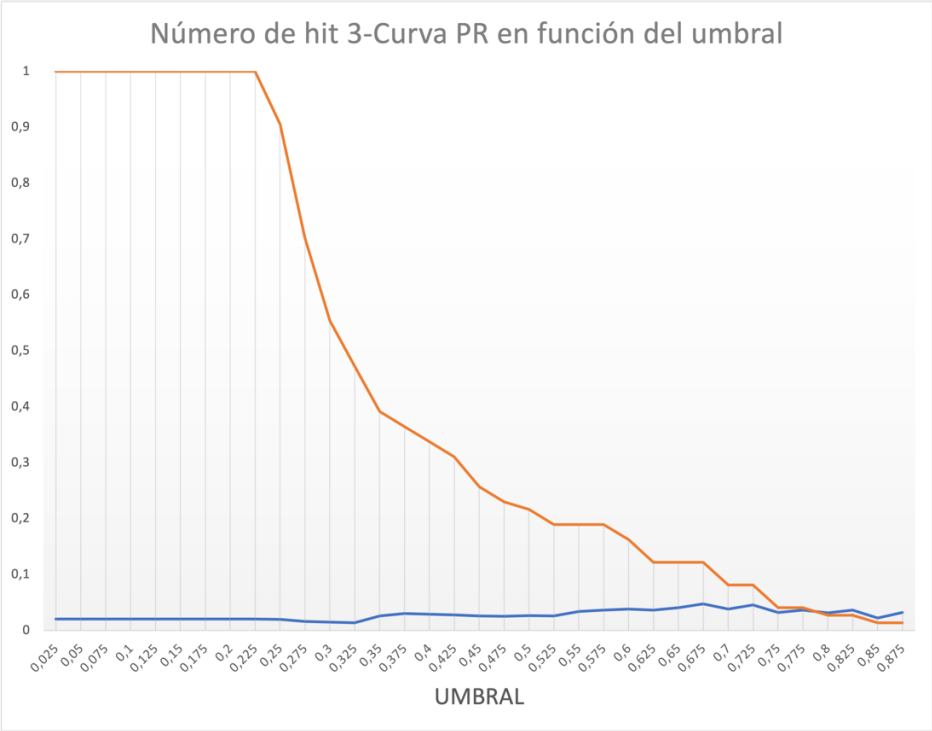
A continuación, se exponen las gráficas y métricas para el conjunto de los diferentes *hits*.

4.2.1 Curva precisión-recall en función del umbral

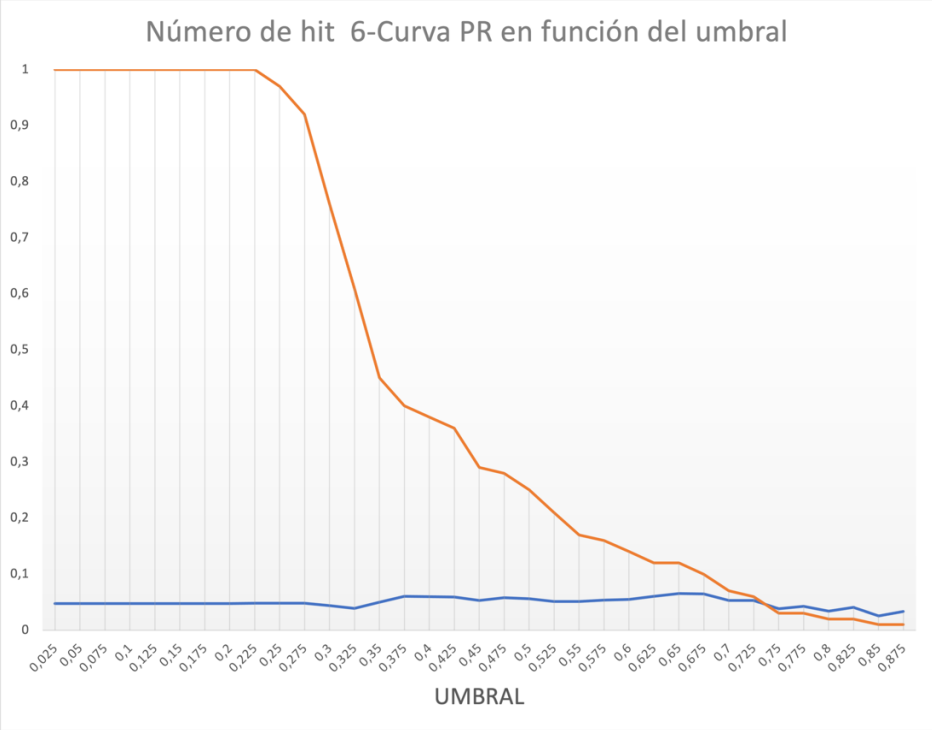
La Figura 33 muestra las gráficas de la curva precisión-*recall* en función del umbral para diferentes conjuntos de datos con el mismo número de *hit*. En ellas se puede observar que el comportamiento no difiere mucho con lo que ocurría cogiendo todo el conjunto de datos que vimos en la anterior sección. Sin embargo, se observa que el *recall* disminuye más lento a medida que aumenta el número de *hits*, ya que al tener un número más alto de *hit* la probabilidad de compra va a ser mayor y con ello el aumento de los verdaderos positivos, en relación con los falsos negativos. En cuanto a la precisión, se puede ver que el valor aumenta, pero sigue fluctuando en torno a un valor, no crece a medida que crece el umbral como debería ocurrir. Como comentamos en la anterior sección, es probable que suceda por el mal comportamiento del modelo de predicción usado.



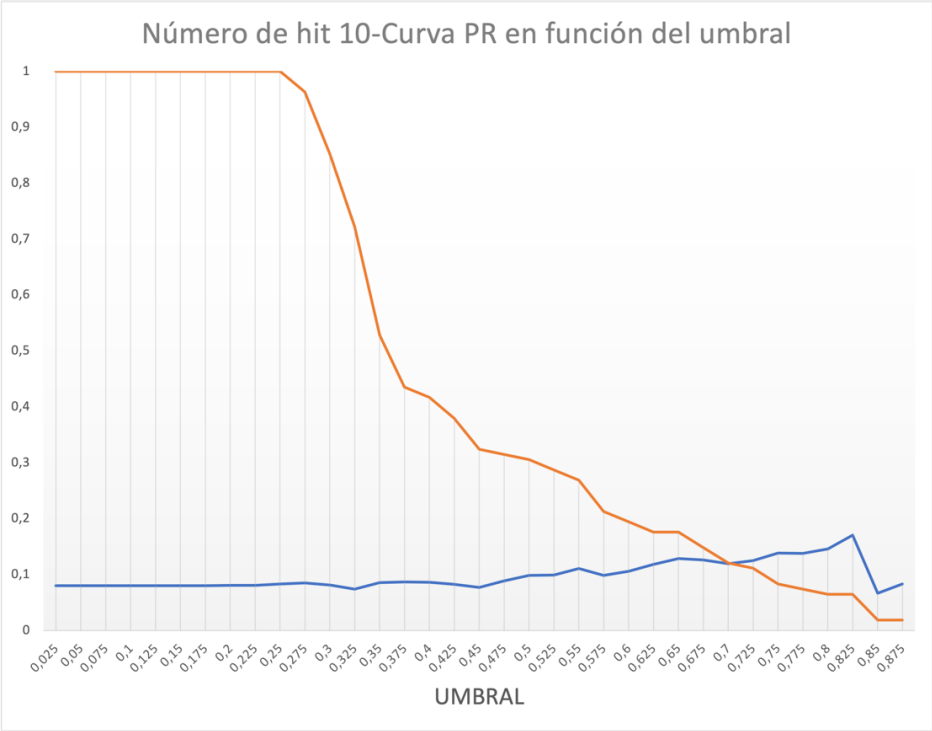
(a) *Hit*=1


Número de hit 3-Curva PR en función del umbral
1
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
0,025 0,05 0,075 0,1 0,125 0,15 0,175 0,2 0,225 0,25 0,275 0,3 0,325 0,35 0,375 0,4 0,425 0,45 0,475 0,5 0,525 0,55 0,575 0,6 0,625 0,65 0,675 0,7 0,725 0,75 0,775 0,8 0,825 0,85 0,875
UMBRAL

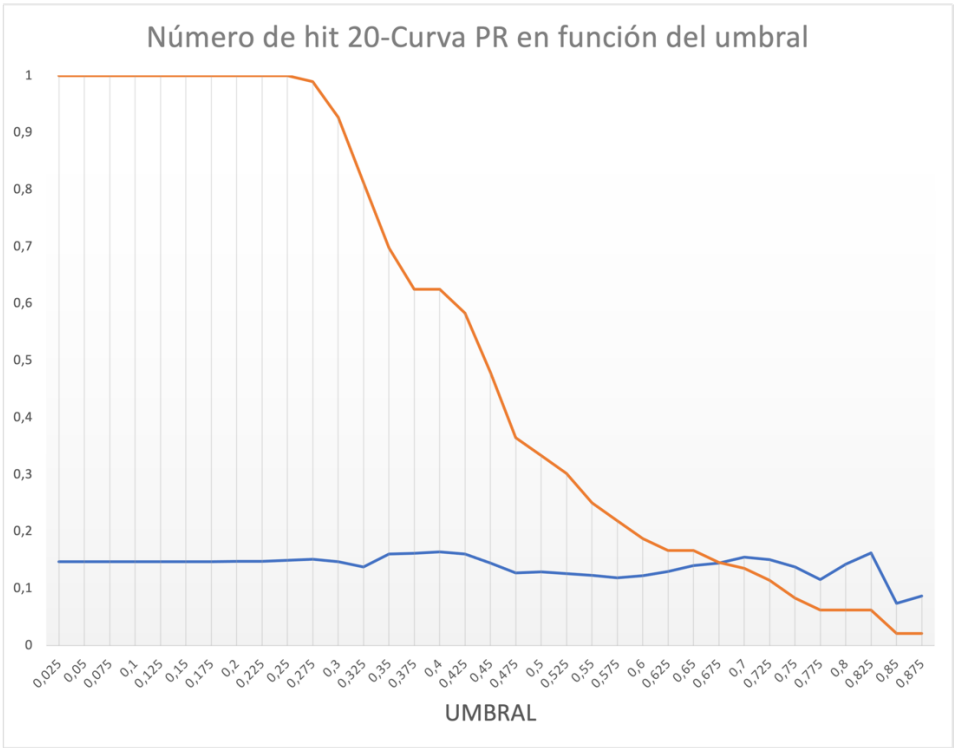
(b) *Hit*=3


Número de hit 6-Curva PR en función del umbral
1
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
0,025 0,05 0,075 0,1 0,125 0,15 0,175 0,2 0,225 0,25 0,275 0,3 0,325 0,35 0,375 0,4 0,425 0,45 0,475 0,5 0,525 0,55 0,575 0,6 0,625 0,65 0,675 0,7 0,725 0,75 0,775 0,8 0,825 0,85 0,875
UMBRAL

(c) *Hit*=6



(d) *Hit*=10

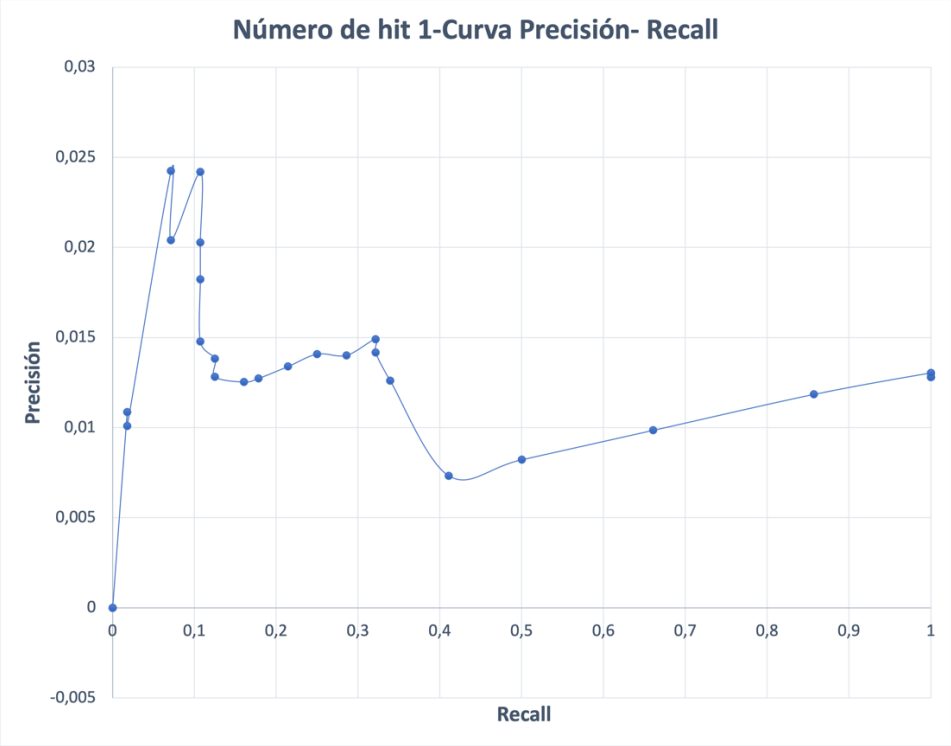


(e) *Hit*=20

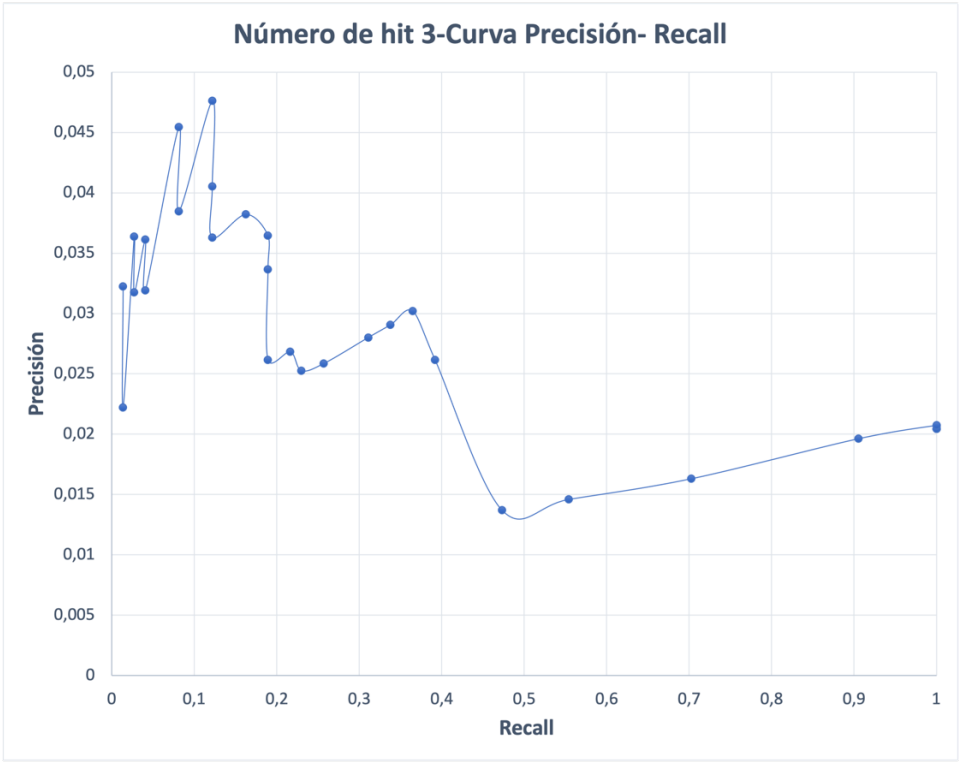
Figura 33- Gráficas precisión-recall en función del umbral para conjuntos de datos con el mismo número de hit.

4.2.2 Curva de precisión-recall

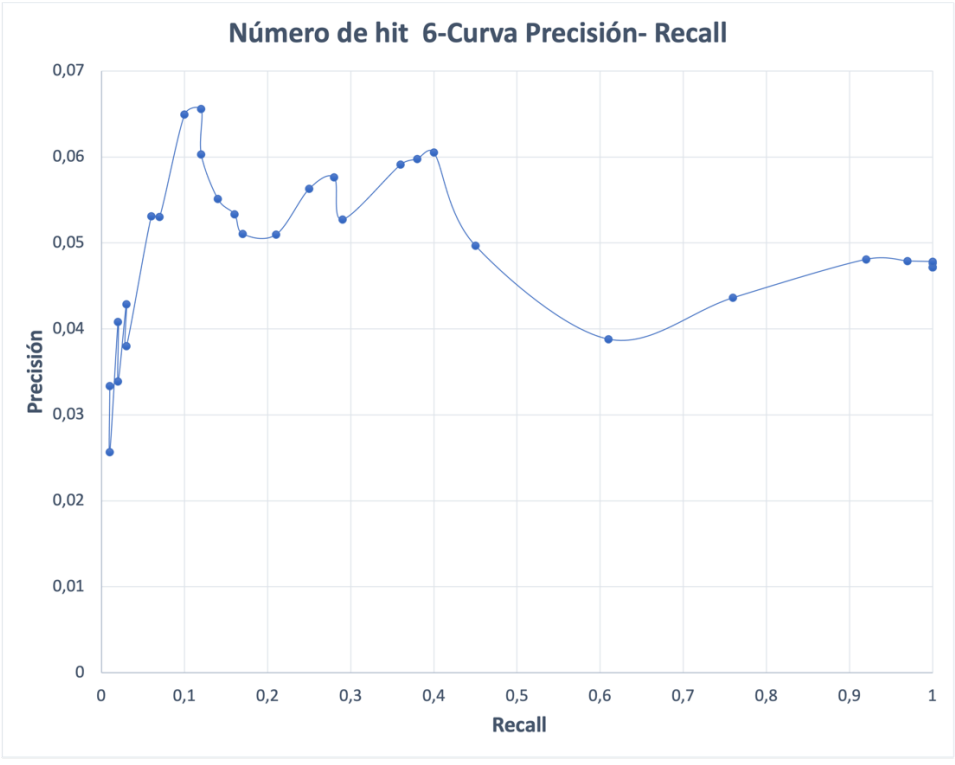
La Figura 34 muestra las gráficas de precisión-*recall* para diferentes conjuntos de datos con el mismo número de *hit*. El *recall* se aprecia que en todos los casos recorre el margen entero desde el máximo hasta el mínimo. Sin embargo, para el valor de la precisión, se observa que sigue habiendo fluctuaciones como en el caso que veíamos para el conjunto completo de *hits*. También se puede ver que el valor de la precisión aumenta conforme aumenta el número de *hit*. Para el conjunto de los primeros *hits* se encuentra en torno al 2%, para los terceros aumenta hasta llegar al 5%, y para conjuntos de datos con numero de *hit* 10 o 20, sobrepasa la barrera del 10%. Esto se debe a que, en proporción, los verdaderos positivos aumentan más rápido que los falsos positivos gracias a que la probabilidad de compra de los productos aumenta al aumentar el número de *hits* que realiza el cliente.



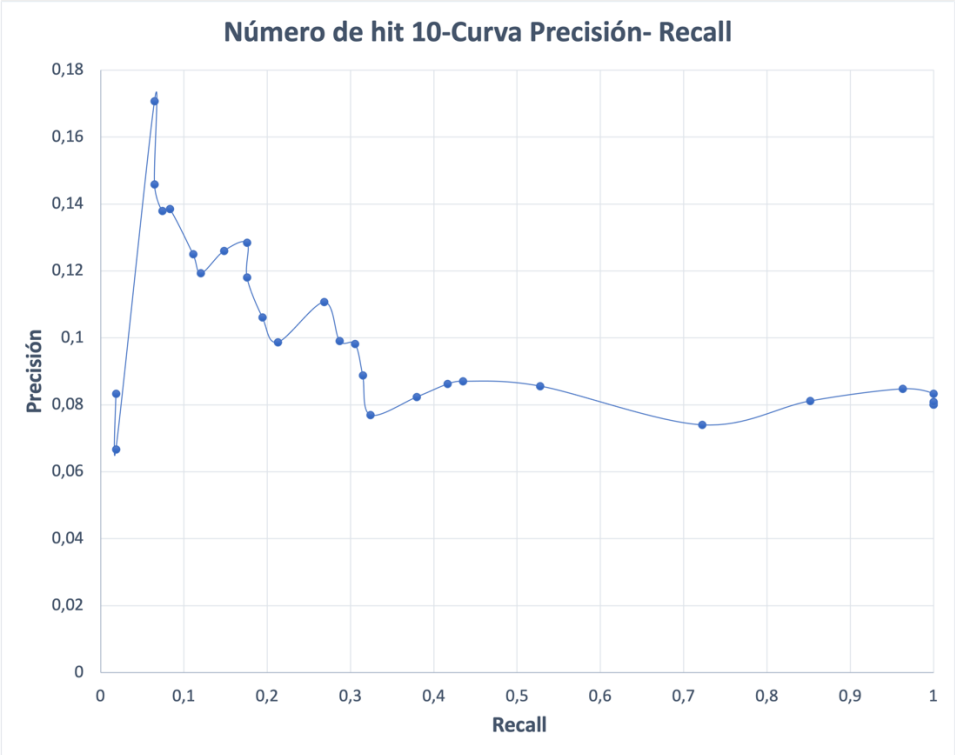
(a) *Hit*=1


Número de hit 3-Curva Precisión- Recall
Precisión
0,05
0,045
0,04
0,035
0,03
0,025
0,02
0,015
0,01
0,005
0
0
0,1
0,2
0,3
0,4
0,5
0,6
0,7
0,8
0,9
1
Recall

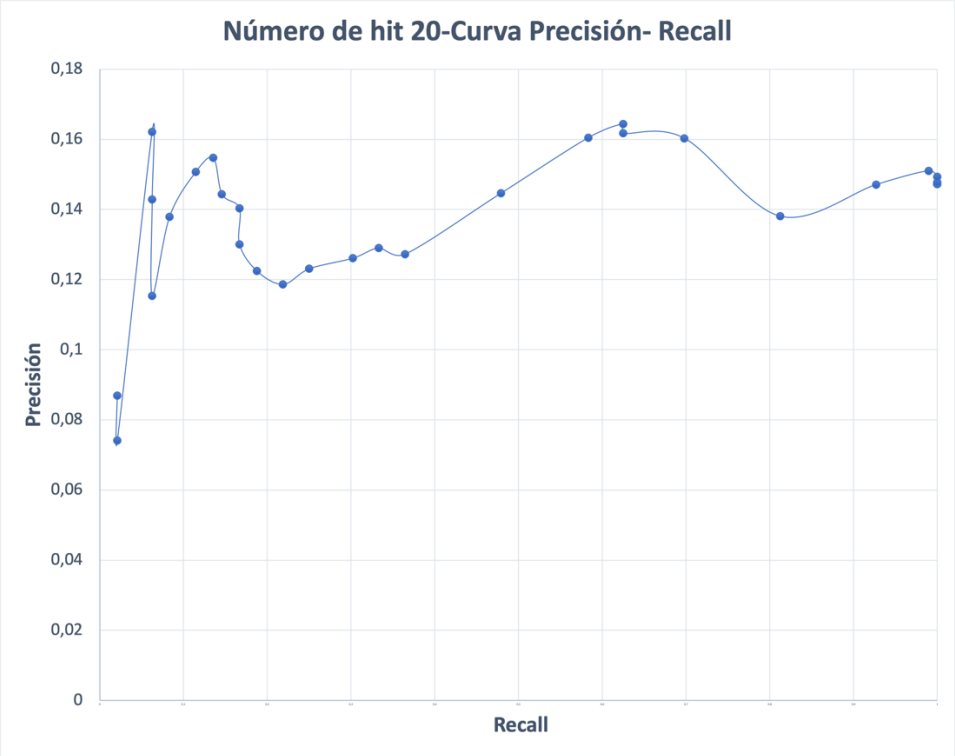
(b) Hit=3


Número de hit 6-Curva Precisión- Recall
Precisión
0,07
0,06
0,05
0,04
0,03
0,02
0,01
0
0
0,1
0,2
0,3
0,4
0,5
0,6
0,7
0,8
0,9
1
Recall

(c) *Hit*=6



(d) *Hit*=10



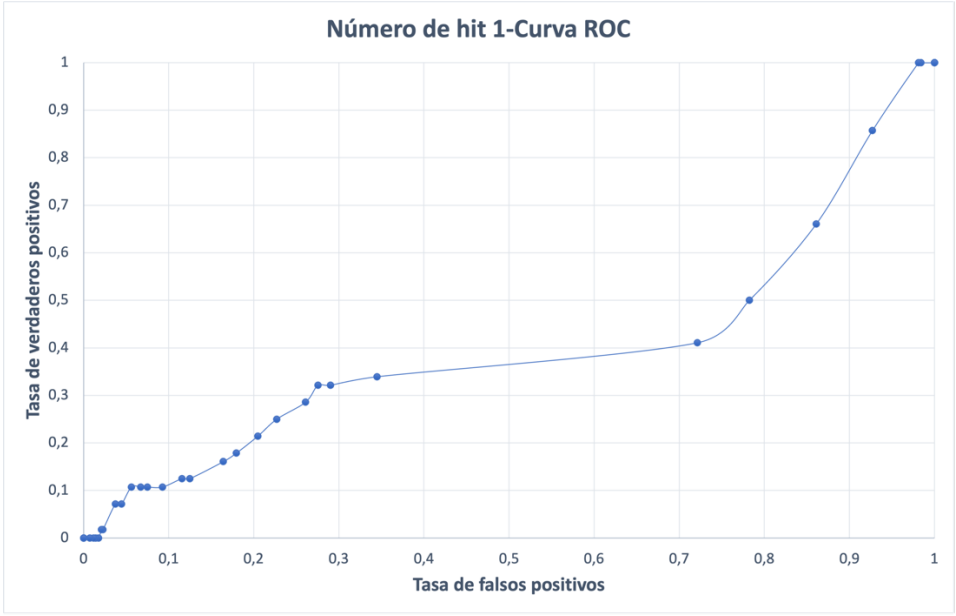
(e) *Hit*=20

Figura 34- Gráficas precisión-recall para conjuntos de datos con el mismo número de hit.

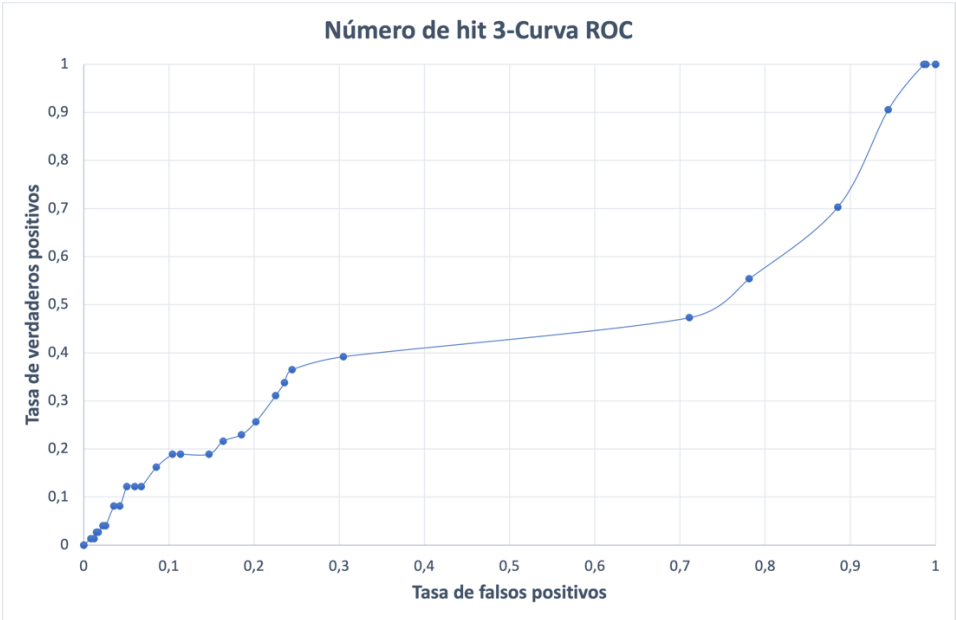
4.2.3 Curva ROC

La Figura 35 muestra la curva ROC para los diferentes conjuntos de datos con el mismo número de *hit*. Se puede apreciar que la curva ROC no se aproxima a un buen modelo de

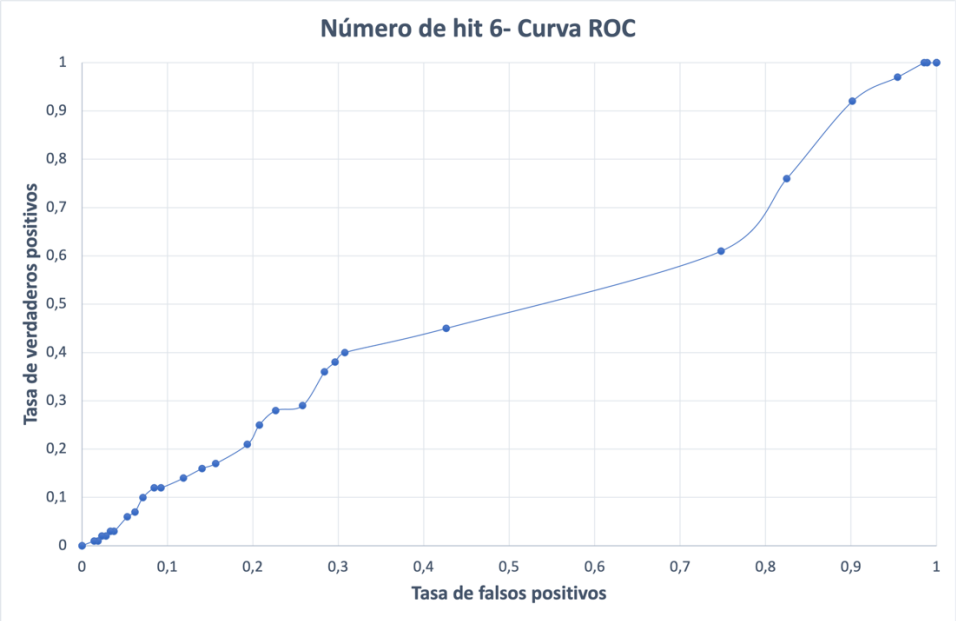
predicción. Como se ha expuesto en los apartados anteriores es probable que haya que elegir otro modelo de predicción o volver a entrenar el modelo para ver si se pueden obtener mejores resultados. Aunque el modelo no obtenga buenos resultados se puede analizar el comportamiento que presenta en función del aumento del número de *hit.* En estas gráficas se observa que la curva ROC mejora en función del número de acciones que realiza el usuario.



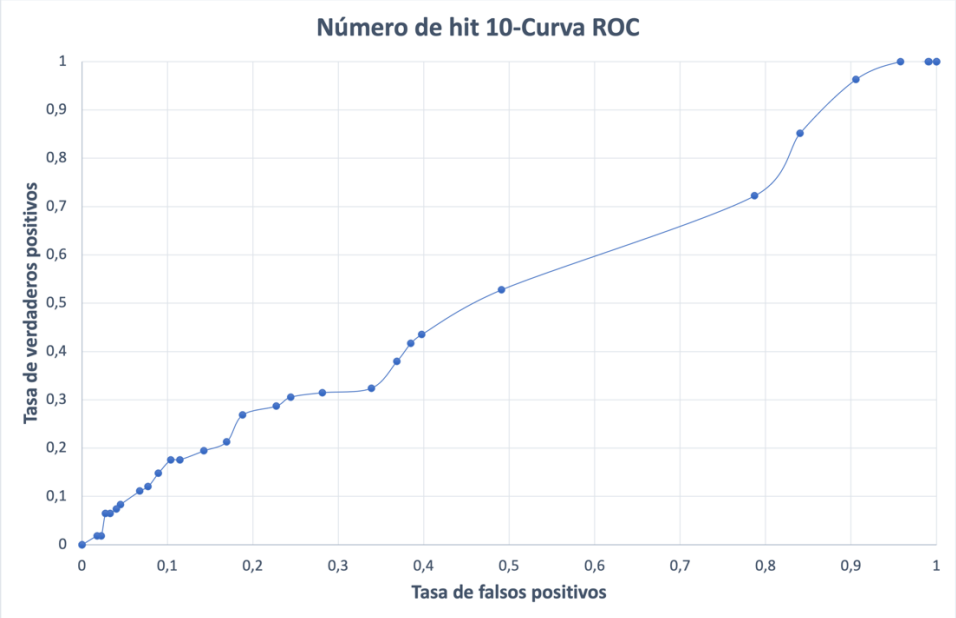
(a) *Hit*=1



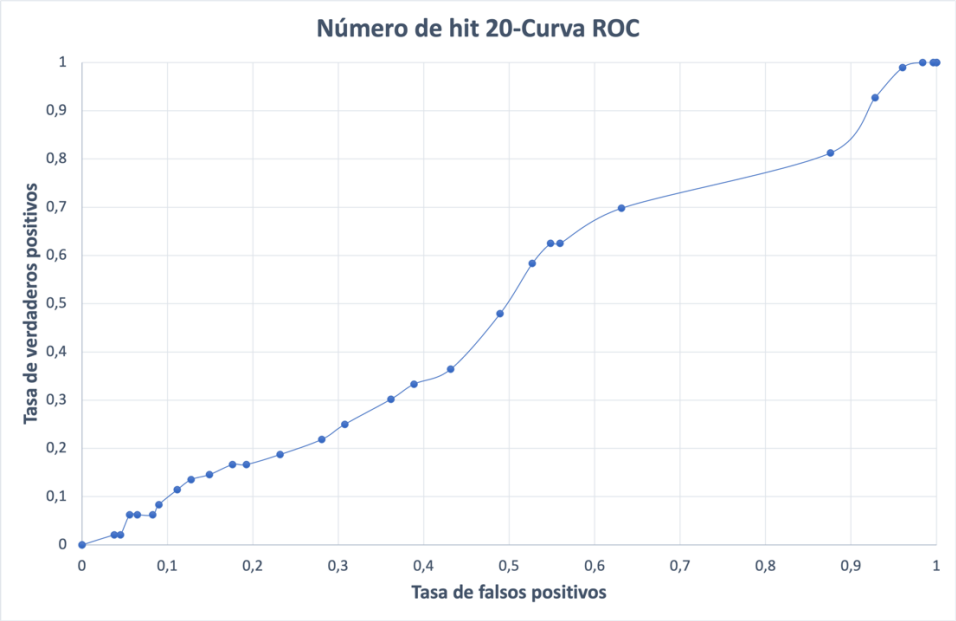
(b) *Hit*=3



(c) *Hit*=6



(d) *Hit*=10



(e) *Hit*=20

Figura 35- Curvas ROC para conjuntos de datos con el mismo número de hit.

Para evaluar la curva ROC, el parámetro más representativo es el área bajo la curva (AUC), que se muestran en la Tabla 2. El área crece conforme el número de acciones del cliente aumenta. Este incremento del AUC se debe a que, al aumentar el número de *hits* del cliente, el número de verdaderos positivos aumenta en relación con los falsos positivos. Sin embargo, este incremento sucede hasta un cierto número de *hit*. Si vemos el valor AUC del conjunto de datos con número de *hit* 20, es menor que el del conjunto de datos con número de *hit* 10. Esto se debe a que no hay suficientes clientes con 20 acciones como para hacer una correcta evaluación. Para solventar el problema se debería coger un conjunto de datos más grande.

	Hits 1	*Hits 3*	*Hits 6*	*Hits 10*	*Hits 20*
AUC	0,3984	0,4449	0,4891	0,5187	0,4913

Tabla 2- Valor AUC de los conjuntos de datos para el mismo número de hit

Como se ha comprobado el sistema de predicción desarrollado no presenta buenos resultados. Para poder solventarlo se deben realizar más pruebas con diferentes modelos de predicción, ya sea cambiando el algoritmo utilizado o modificando algunos de sus parámetros. También se puede volver a entrenar el modelo con datos más recientes o que tengan más transacciones, ya que el número de verdaderos positivos es muy pequeño frente a los demás parámetros.

5 Conclusiones y líneas futuras

En este capítulo se van a exponer las conclusiones del proyecto desarrollado y las posibles líneas futuras.

Como se ha comentado en el primer capítulo, el principal objetivo del trabajo ha sido la implementación de un sistema de predicción de compra para clientes de una página web determinada. Para ello, se partió de un *dataset* de Google Analytics con las sesiones de los clientes de la web junto las transacciones que realizaban, así como los *hits* o acciones de cada sesión. Con los datos almacenados se ha hecho un sistema simulando el tiempo real, contando con la lectura de los datos, la API flask y la predicción de probabilidad de compra con mecanismos de inteligencia artificial integrados en el modelo.

En este proyecto se ha visto que se pueden analizar los datos de los clientes en tiempo real, sin tener que recoger una sesión entera de cliente para poder predecir si va a comprar. Es decir, ha servido para poder ver con anterioridad a la finalización de la sesión que un cliente tiene intenciones de comprar un determinado producto.

Por ejemplo, un cliente que está navegando en una página web y ha realizado 50 hits es probable que compre el producto visitado. Con el sistema antiguo se haría una predicción de compra tras finalizar su sesión, en cambio, con el nuevo sistema desarrollado se puede predecir la probabilidad de compra en cada *hit*, y así poder predecir que va a comprar antes de que termine su sesión.

Se ha visto en los resultados obtenidos que el modelo de predicción tiene bastantes problemáticas ya que los resultados no son los que se esperaban en un principio. Al comparar el sistema antiguo con el nuevo se ha visto que el modelo de predicción no es el adecuado para hacer predicciones en tiempo real. Esto nos lleva a pensar en las posibles causas de que el funcionamiento no sea el correcto. Como causa principal se piensa que el modelo no se ha entrenado correctamente o lo suficiente, por tanto, deben hacerse más pruebas para poder solventarlo. Otra causa puede ser que el algoritmo de inteligencia artificial utilizado para en el modelo de predicción antiguo no sea eficiente para realizar las predicciones individuales de cada *hit*. Y la última causa pensada ha sido la falta de datos para realizar el testeo. Se vio en los resultados que las transacciones reales de los clientes eran escasas y esto hacía que los verdaderos positivos también lo fueran.

Las líneas futuras planteadas son principalmente las mejoras del sistema. Se debe hacer un reentrenamiento del modelo de predicción y verificar sus resultados. Si los resultados siguen siendo malos se plantearía la posibilidad de cambiar el algoritmo de inteligencia artificial utilizado. Además, se podría aumentar el conjunto de datos de testeo para ver si esa puede ser la causa de los malos resultados.

Una vez que el sistema funcione correctamente y muestre los resultados esperados, se podría continuar con la creación de un *dashboard* en el que las empresas pudiesen obtener datos de compras de los clientes que visitan sus páginas web. Con estas métricas que obtiene la empresa podrían personalizar las campañas publicitarias para los clientes en tiempo real.

6 Bibliografía

[1] M. Perez, «Conceptodefinicion,» 28 julio 2021. [En línea]. Available: https://conceptodefinicion.de/comercio/. [Último acceso: 18 septiembre 2022].

[2] p. jaja, «WebWngage,» WebWngage, 23 junio 2022. [En línea]. Available: https://webengage.com/blog/hyper-personalization-marketing-future/. [Último acceso: 14 julio 2022].

[3] RAE, «Real Academia Española,» [En línea]. Available: https://www.rae.es/dpd/m%C3%A1rquetin.

[4] Charteres Institute of Marketing (CI;), «A brief summary of marketing and how it works,» *Marketing and the 7Ps,* p. 12, 2015.

[5] AMA, «American Marketing Association,» American Marketing Association, [En línea]. Available: https://www.ama.org/. [Último acceso: 14 Julio 2022].

[6] RD STATION, «rdstation,» RD STATION, [En línea]. Available: https://www.rdstation.com/es/marketing-digital/. [Último acceso: 14 julio 2022].

[7] J. Santaella, «economia3,» economia3, 12 Septiembre 2021. [En línea]. Available: https://economia3.com/historia-marketing-digital-hasta-nuestros-dias-fechas-clave/. [Último acceso: 14 julio 2022].

[8] J. Rocamora, «marketing4ecommerce,» marketing4ecommerce, 11 mayo 2016. [En línea]. Available: https://marketing4ecommerce.net/web-4-0-la-revolucion-ecommerce/. [Último acceso: 14 julio 2022].

[9] R. Garcia, «ecommercees,» ecommercees, 13 mayo 2022. [En línea]. Available: https://ecommercees.com/la-hiper-personalizacion-en-el-ecommerce/. [Último acceso: 14 julio 2022].

[10] marketingdirecto, «marketingdirecto,» marketingdirecto, 24 febrero 2022. [En línea]. Available: https://www.marketingdirecto.com/marketing-general/marketing/hiperpersonalizacion-campanas-aumenta-impacto-fidelizacion. [Último acceso: 14 julio 2022].

[11] T. Lebo, «Convinceandconvert,» Convinceandconvert, [En línea]. Available: https://www.convinceandconvert.com/digital-marketing/hyper-personalization/. [Último acceso: 14 julio 2022].

[12] N. Rivas, «Seosve,» Seosve, [En línea]. Available: https://www.seosve.com/introduccion-al-marketing-digital/. [Último acceso: 19 julio 2022].

[13] M. Acibero, «godaddy,» godaddy, 19 julio 2021. [En línea]. Available: https://es.godaddy.com/blog/que-es-posicionamiento-seo/. [Último acceso: 19 julio 2022].

[14] L. Cardona, «cyberclick,» cyberclik, 2019 marzo 19. [En línea]. Available: https://www.cyberclick.es/numerical-blog/loyalty-marketing-el-complemento-ideal-para-tu-estrategia-de-seo. [Último acceso: 20 julio 2022].

[15] H. M. Al-Barhamtoshy y . F. E. Eassa, «A Data Analytic Framework for Unstructured Text,» *Life Science Journal,* 2014.

[16] Universidad de Alcalá, «La importancia de que haya profesionales ne Big Data,» Universidad de Alcalá, 2022. [En línea]. Available: https://www.master-data-scientist.com/importancia-curso-big-data-madrid/. [Último acceso: 20 julio 2022].

[17] Azure, «Azure,» Azure, 2022. [En línea]. Available: https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-machine-learning-platform/#benefits. [Último acceso: 20 julio 2022].

[18] J. L. Gonzalez, «medium,» medium, 8 febrero 2018. [En línea]. Available: https://medium.com/soldai/tipos-de-aprendizaje-autom%C3%A1tico-6413e3c615e2. [Último acceso: 20 julio 2022].

[19] H. Tran, «A SURVEY OF MACHINE LEARNING AND DATA MINING TECHNIQUES USED IN MULTIMEDIA SYSTEM,» Texas, 2019.

[20] D. A. Pisne y D. M. Schnyer, «Support vector machine,» de *Machine Learning-Methods and Applications to Brain Disorders*, Austin, Academic Press, 2019, p. 20.

[21] J. Ali, N. Ahmad y R. Khan, «Random Forests and Decision Trees,» *International Journal of Computer Science Issues,* vol. 9, nº 3, pp. 272-278, 2012.

[22] JavaTpoint, «JavaTpoint,» Javatpoint, [En línea]. Available: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm. [Último acceso: 21 julio 2022].

[23] . L. Wu, Y. Yuan y . X. Zhang, «Gini-Impurity Index Analysis,» *TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY,* vol. 16, pp. 3154-3169, 2021.

[24] C. F. Institute, «Corporate Finance Institute,» Corporate Finance Institute, 1 septiembre 2021. [En línea]. Available: https://corporatefinanceinstitute.com/resources/knowledge/other/random-forest/. [Último acceso: 21 julio 2022].

[25] J. M. Heras, «iartificial.net,» iartificial.net, 18 septiembre 2020. [En línea]. Available: https://www.iartificial.net/random-forest-bosque-aleatorio/#%C2%BFQue_es_un_Random_Forest. [Último acceso: 21 julio 2022].

[26] Seoungjuhong, «Seoungjuhong Marker,» Seoungjuhong, 17 enero 2017. [En línea]. Available: https://seongjuhong.com/2021-01-17pm-ensemble-bagging-and-boosting/. [Último acceso: 21 julio 2022].

[27] Unioviedo, «El algoritmo k-means aplicado a clasificación y procesamiento de imágenes,» Unioviedo, [En línea]. Available: https://www.unioviedo.es/compnum/laboratorios_py/kmeans/kmeans.html#El-algoritmo-k-means. [Último acceso: 25 julio 2022].

[28] Profesordata, «Profesordata.com,» Profesordata, 7 agosto 2020. [En línea]. Available: https://profesordata.com/2020/08/07/evaluando-los-modelos-de-clasificacion-en-aprendizaje-automatico-la-matriz-de-confusion-claramente-explicada/. [Último acceso: 25 julio 2022].

[29] A. Kulkarnii, D. Chong y F. A. Batarseh, «Foundations of data imbalance and solutions for a data democracy,» de *Data Democracy*, Washington, Academic press, 2020, pp. 83-106.

[30] R. D. Sipio, «towardsdatascience,» towardsdatascience, 31 mayo 2021. [En línea]. Available: https://towardsdatascience.com/a-quick-guide-to-auc-roc-in-machine-learning-models-f0aedb78fbad. [Último acceso: 28 julio 2022].

[31] J. Ramírez, «medium,» medium, 19 julio 2018. [En línea]. Available: https://medium.com/bluekiri/curvas-pr-y-roc-1489fbd9a527. [Último acceso: 28 julio 2022].

[32] N. Kumar Sehgal y P. Chandra P. Bhatt, Cloud Computing, Concepts and Practices, Santa Clara: Springer, 2018.

[33] RedHat, «RedHat,» 15 marzo 2018. [En línea]. Available: https://www.redhat.com/es/topics/cloud-computing/public-cloud-vs-private-cloud-and-hybrid-cloud#:~:text=Hay%20cuatro%20tipos%20principales%20de,software%20como%20servicio%20(SaaS).. [Último acceso: 1 agosto 2022].

[34] R. Kumar, «Wpoven,» 22 abril 2022. [En línea]. Available: https://www.wpoven.com/blog/cloud-market-share/. [Último acceso: 2 agosto 2022].

[35] E. Jones, «kinsta,» 11 agosto 2021. [En línea]. Available: https://kinsta.com/. [Último acceso: 2 agosto 2022].

7 Anexos

7.1 Anexo 1

En este anexo se van a exponer las figuras correspondientes a los códigos de Python realizados durante el proyecto. Se han realizado dos ficheros, el fichero donde se encuentra la API, denominado *api_model.py* y el fichero donde se encuentra la lectura de los datos y la subida de los mismo a la API denominado *run.py*.

7.1.1 Fichero run.py

El fichero *run.py* se muestra en la Figura 36 donde se pueden ver las dependencias de las librerías, como se realiza la conexión con Google Cloud, la sentencia SQL utilizada para obtener los datos, y la subida de los mismo a la API flask a través de una petición POST.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Jul  4 16:56:51 2022

@author: raulbarbaalonso
"""

import os
import requests
import json
from google.cloud import bigquery

# credenciales de google cloud
os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = "bq-luce-
bd4bd5286194.json"
# cliente de bigQeury
client = bigquery.Client()
while (1):
    # query para obtención de una fila de la tabla de datos: API
request
    query_job = client.query(
        """SELECT
      t.visitorId             AS visitorId,
      t.visitNumber           AS visitNumber,
      t.visitId               AS visitId,
      t.visitStartTime        AS visitStartTime,
      t.date                  AS date,
      t.totals                AS totals,
      t.hits                  AS hits,
      t.fullVisitorId         AS fullVisitorId,
      t.userId                AS  userId,
      t.clientId              AS clientId,
      t.channelGrouping       AS channelGrouping,
      t.socialEngagementType  AS  socialEngagementType,
    FROM `bq-luce.archiever.ga_sessions` AS t
    ORDER BY RAND() LIMIT 1""")

    # resultado de la query que se ha realizado a BigQuery
    results = query_job.result()  # Waits for query to finish

    # Conversión de los datos de una fila a formato JSON
    records = [dict(row) for row in query_job]
    result_json = json.dumps(str(records))
    #print(result_json)
```

```
    #print(result_json)
    # Uri base del servidor Flask
    baseUrl = 'http://localhost:5000'
    #baseUrl='https://app-achiever-dey6vjv7bq-ey.a.run.app'
    # Cabecera de json para realizar la petición
    headers = {'Content-Type': 'application/json; charset=utf-8'}
    # petición POST para el envío de datos al servidor
    r = requests.post(baseUrl + "/datos",
data=result_json,headers=headers)
    print(r)
```

Figura 36- Fichero run.py con el código de lectura de los datos.

7.1.2 Fichero api_model.py

El código generado en el fichero api_model.py se muestra en la Figura 37 donde se puede encontrar la definición de la API flask, la recogida de los datos, la petición SQL realizada a la plataforma de Google Cloud en la que se evalúa el modelo de predicción de probabilidad de compra, y en las últimas líneas se puede ver el guardado de los resultados en un fichero *csv*.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Jul  4 16:56:51 2022

@author: raulbarbaalonso
"""
from flask import Flask, request, jsonify
import os
import requests
import json
import ast
from google.cloud import bigquery
from datetime import datetime
import pandas as pd
import csv

# Definición de la aplicación
app = Flask(__name__)


# Recurso raiz
@app.route('/')
def raiz():
    return {"Directorio": "raiz"}


# recurso datos a donde subimos los datos de la BigQuery
@app.route("/datos", methods=['GET', 'POST'])
def post():
    content: object = request.json

    # credenciales de google cloud
    os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = "bq-luce-
bd4bd5286194.json"
    # cliente de bigQuery
    client = bigquery.Client()

    # escapamos la cadena content del primer y último caracter
    content = content[1:-1]
    # reemplazamos las comillas simples por las dobles
```

```
    data = content.replace("'", "\"")
    # pasamos la cadena a un diccionario de python para poder coger
cada uno de los parámetros
    d = ast.literal_eval(data)


    print("Id del cliente: " + d["clientId"])
    # id del cliente
    id_cliente = d["clientId"]

    # generamos el formato de fecha adecuado para introducirlo en la
query
    d["date"] = datetime.strptime(d["date"], "%Y%m%d").date()
    fecha = d["date"]
    print(fecha)
    visit_Id = d["visitId"]
    print("Id de la visita: " + str(visit_Id))
    hits = d["hits"]

    for hit in hits:
        hit_number = hit["hitNumber"]
        print("Número de hit: " + str(hit_number))

        hour = hit["hour"]
        minute = hit["minute"]
        print("Hora del hit:" + str(hour) + ":" + str(minute))

        query = f"""DECLARE client STRING DEFAULT "{id_cliente}";
        DECLARE predict_date DATE DEFAULT "{fecha}";
        DECLARE hit_date DATETIME DEFAULT DATETIME(PARSE_DATE('%Y-%m-
%d', "{fecha}"), TIME({hour}, {minute}, 0));
        DECLARE visit_Id INT64 DEFAULT {visit_Id};
        DECLARE hit_number INT64 DEFAULT {hit_number};

# Query:
WITH table1 AS (
    SELECT
        sessions.visitId              AS visitId,
        sessions.clientId             AS clientId,
        sessions.fullVisitorId        AS fullVisitorId,
        PARSE_DATE("%Y%m%d",sessions.date)  AS fecha,
        sessions.totals.pageviews     AS total_pageviews,
        h.page.pagePath               AS page,
        SUM(COALESCE(tp.tiempo_pagina, 0))   AS tiempo_pagina,
        COALESCE(h.transaction.transactionRevenue, 0) AS valor,
        h.ecommerceaction.action_type AS tipo_accion,
        hits_product.v2ProductName    AS producto,
        hits_product.v2ProductCategory        AS categoria_producto,
        carrito.producto_carrito      AS producto_carrito,
        carrito.categoria_carrito     AS categoria_carrito,

    FROM `bq-luce.archiever.ga_sessions` AS sessions
    LEFT JOIN UNNEST(hits) AS h
    LEFT JOIN UNNEST(h.product) AS hits_product
    LEFT JOIN (
        SELECT
            ga.visitId                AS visitId,
            ga.fullVisitorId          AS fullVisitorId,
            ga.totals.timeOnSite      AS duracion_sesion,
            h.page.pagePath           AS page,
            h.time                    AS time,
```

```
            h.type                    AS tipo,
            CASE
                WHEN (LAG(h.time, -1) OVER (ORDER BY ga.visitId,
h.time)) - h.time > 0 THEN (LAG(h.time, -1) OVER (ORDER BY ga.visitId,
h.time)) - h.time
                WHEN ga.totals.timeOnSite*1000 - h.time > 0 THEN
ga.totals.timeOnSite*1000 - h.time
                ELSE 0 END   AS tiempo_pagina
        FROM `bq-luce.archiever.ga_sessions` AS ga
        LEFT JOIN UNNEST(hits) AS h
        WHERE
            ga.clientId = client AND h.type LIKE 'PAGE'AND ga.date
BETWEEN (SELECT FORMAT_DATETIME("%Y%m%d", DATE_SUB(predict_date,
INTERVAL 15 DAY))) AND (SELECT FORMAT_DATETIME("%Y%m%d",
predict_date)) AND ( (DATETIME(PARSE_DATE('%Y%m%d', ga.date),
TIME(h.hour, h.minute, 0)) < hit_date) OR
                  (DATETIME(PARSE_DATE('%Y%m%d', ga.date),
TIME(h.hour, h.minute, 0)) = hit_date AND visitId = visit_Id AND
hitNumber <= hit_number))
        GROUP BY ga.visitId, ga.fullVisitorId, h.page.pagePath,
ga.totals.timeOnSite, h.type, h.time
    ) AS tp ON
        h.page.pagePath = tp.page
        AND sessions.visitid = tp.visitId
        AND sessions.fullVisitorId = tp.fullVisitorId
        AND h.time = tp.time
        AND h.type = tp.tipo

    LEFT JOIN (
        SELECT
            aux_table.visitId          AS visitId,
            aux_table.fullVisitorId    AS fullVisitorId,
            aux_table.producto         AS producto_carrito,
            aux_table.categoria        AS categoria_carrito
        FROM (
            SELECT
                ga.visitId             AS visitId,
                ga.fullVisitorId       AS fullVisitorId,
                hp.v2ProductName       AS producto,
                hp.v2ProductCategory   AS categoria,
                MAX(h.ecommerceaction.action_type) AS max_action_type

            FROM `bq-luce.archiever.ga_sessions` AS ga
            LEFT JOIN UNNEST(hits) AS h
            LEFT JOIN UNNEST(h.product) AS hp
            WHERE
                ga.clientId = client
                AND (h.ecommerceaction.action_type = '6' OR
h.ecommerceaction.action_type = '3')
                AND ga.date BETWEEN
                    (SELECT FORMAT_DATETIME("%Y%m%d",
DATE_SUB(predict_date, INTERVAL 15 DAY)))
                    AND (SELECT FORMAT_DATETIME("%Y%m%d",
predict_date))
                AND ( (DATETIME(PARSE_DATE('%Y%m%d', ga.date),
TIME(h.hour, h.minute, 0)) < hit_date) OR
                    (DATETIME(PARSE_DATE('%Y%m%d', ga.date),
TIME(h.hour, h.minute, 0)) = hit_date AND visitId = visit_Id AND
hitNumber <= hit_number))
            GROUP BY ga.visitId, ga.fullVisitorId, hp.v2ProductName,
hp.v2ProductCategory
```

```
        ) AS aux_table
        WHERE aux_table.max_action_type = '3'
        GROUP BY aux_table.visitId, aux_table.fullVisitorId,
aux_table.producto, aux_table.categoria

    ) AS carrito ON
        sessions.visitId = carrito.visitId
        AND sessions.fullVisitorId = carrito.fullVisitorId
        AND hits_product.v2ProductName = carrito.producto_carrito
        AND hits_product.v2ProductCategory = carrito.categoria_carrito
        AND  h.ecommerceaction.action_type = '3'
    WHERE
        sessions.clientId = client
        AND h.ecommerceaction.action_type != '0'
        AND h.type = 'PAGE'
        AND sessions.totals.timeOnSite > 0
        AND sessions.totals.timeOnSite IS NOT NULL
        AND hits_product.v2ProductName IS NOT NULL
        AND hits_product.v2ProductCategory IS NOT NULL
        AND hits_product.v2ProductCategory != '(not set)'
        AND sessions.date BETWEEN
            (SELECT FORMAT_DATETIME("%Y%m%d", DATE_SUB(predict_date,
INTERVAL 15 DAY)))
            AND (SELECT FORMAT_DATETIME("%Y%m%d", predict_date))
        AND ( (DATETIME(PARSE_DATE('%Y%m%d', sessions.date),
TIME(h.hour, h.minute, 0)) < hit_date) OR
             (DATETIME(PARSE_DATE('%Y%m%d', sessions.date),
TIME(h.hour, h.minute, 0)) = hit_date AND sessions.visitId = visit_Id
AND hitNumber <= hit_number))
    GROUP BY
        sessions.visitId,
        sessions.clientId,
        sessions.fullVisitorId,
        sessions.date,
        sessions.totals.pageviews,
        h.page.pagePath,
        h.time,
        h.transaction.transactionRevenue,
        h.ecommerceaction.action_type,
        hits_product.v2ProductName,
        hits_product.v2ProductCategory,
        producto_carrito,
        categoria_carrito
),
table2 AS (
   SELECT
        table1.clientId,
        table1.visitId,
        AVG(table1.tiempo_pagina) AS tiempo_pag_medio
    FROM table1
    WHERE table1.tiempo_pagina > 0
    GROUP BY table1.clientId, table1.visitId
),
table3 AS (
    SELECT
        table1.*,
        (CASE WHEN table1.tiempo_pagina = 0 THEN
table2.tiempo_pag_medio ELSE table1.tiempo_pagina END) AS
tiempo_pag_final
    FROM table1
    INNER JOIN table2
```

```
    ON table1.clientId = table2.clientId AND table1.visitId =
table2.visitId
),
sesiones_totales AS (
    SELECT
        table3.clientId AS clientId,
        table3.fecha    AS fecha_visita
    FROM table3
    WHERE table3.fecha = predict_date
    GROUP BY table3.clientId, table3.fecha
)
SELECT
    fecha_visita,
    categoria_producto,
    clientId,
    num_sesiones,
    num_pageviews,
    dias_transcurridos,
    avg_page_time,
    num_elem_carrito,
    num_elem_comprados,
    predicted_label                          AS prediction,
    predicted_label_probs[OFFSET(0)].prob    AS prob_compra,
    predicted_label_probs[OFFSET(1)].prob    AS prob_no_compra
FROM
    ML.PREDICT(MODEL `bq-luce.archiever.cdl_model`, (
        SELECT
            st.fecha_visita           AS fecha_visita,
            t.categoria_producto      AS categoria_producto,
            t.clientId                AS clientId,
            COUNT(DISTINCT t.visitId) AS num_sesiones,
            COUNT(*)                  AS num_pageviews,
            DATE_DIFF(st.fecha_visita, MAX(t.fecha), DAY)
AS dias_transcurridos,
            AVG(t.tiempo_pag_final)   AS avg_page_time,
            SUM(CASE WHEN t.categoria_producto = t.categoria_carrito
THEN 1 ELSE 0 END)                     AS num_elem_carrito,
            MAX(CASE WHEN t.tipo_accion = '6' and t.valor > 0 THEN 1
ELSE 0 END)                            AS num_elem_comprados
        FROM sesiones_totales AS st
        INNER JOIN
            (SELECT *
            FROM table3
            WHERE
                DATE_SUB(predict_date, INTERVAL 15 DAY) <=table3.fecha
                AND table3.fecha <= predict_date) AS t
        ON st.clientId = t.clientId
        WHERE
            DATE_SUB(st.fecha_visita, INTERVAL 15 DAY) <= t.fecha
            AND t.fecha <= st.fecha_visita
        GROUP BY st.fecha_visita, t.categoria_producto, t.clientId
        )
        );"""
        query_job = client.query(query)

        # resultado de la query que se ha realizado a BigQuery
        results = query_job.result()  # Waits for query to finish

        records = [dict(row) for row in results]
```

```
        print("Tamaño del records: ", len(records))
        lon_records=len(records)
        while lon_records>0:
            lon_records = lon_records - 1

categoria_producto=records[lon_records]['categoria_producto']
            prob_compra=records[lon_records]['prob_compra']
            print("Probabilidad de compra: ", str(prob_compra))
            print(records)

            with open('ejemplo.csv', 'a+') as csvfile:
                fieldnames = ['id_cliente',
'id_visita','categoria_producto','numero_hit', 'probabilidad_compra']
                writer = csv.DictWriter(csvfile,fieldnames=fieldnames)

                writer.writerow({'id_cliente': id_cliente,
'id_visita':visit_Id,'categoria_producto':categoria_producto,'numero_h
it':hit_number, 'probabilidad_compra': prob_compra})

        result_json = json.dumps(str(records))

    return result_json
```

Figura 37- Fichero api_model con el código referente a la API flask

7.1.3 Dockerfile

También se va a mostrar el contenido del *Dockerfile*, que como se mencionó, es el fichero en el que se describen las instrucciones que se van a realizar para generar la imagen Docker. En la Figura 38 se muestra el código relativo a dichas instrucciones.

```
# syntax=docker/dockerfile:1

FROM python:3.8-slim-buster

ENV FLASK_APP api_model
ENV FLASK_ENV development

WORKDIR /Desktop/flask-docker-app
ENV PORT='8080'
ENV HOST='0.0.0.0'
# ENV GOOGLE_APPLICATION_CREDENTIALS='/Desktop/flask-docker-app/bq-
luce-bd4bd5286194.json'
EXPOSE 8080
COPY ./ ./

COPY requirements.txt requirements.txt
RUN pip3 install -r requirements.txt

COPY . .

CMD [ "python3", "-m" , "flask", "run", "--host=0.0.0.0", "--
port=8080" ]
```

Figura 38- Código del fichero Dockerfile para la generación de la imagen Docker.

7.1.4 Fichero csv

El contenido del fichero *csv* se aprecia en la Figura 39. En ella observamos los resultados obtenidos tras la ejecución del modelo de predicción de compra. En concreto se han obtenido los datos del identificador del cliente, el identificador de la visita, la categoría del producto, número de *hit* correspondiente y probabilidad de compra.

```
555682726.1655081031,1655081085,Literatura,1,0.24020572006702423
555682726.1655081031,1655081085,Literatura,2,0.26881691813468933
555682726.1655081031,1655081085,Literatura,3,0.26881691813468933
555682726.1655081031,1655081085,Literatura,4,0.26881691813468933
555682726.1655081031,1655081085,Romántica y
erótica,5,0.26881691813468933
555682726.1655081031,1655081085,Literatura,5,0.26881691813468933
1179060613.1655578721,1655578720,Psicología y
Pedagogía,2,0.29078415036201477
1179060613.1655578721,1655578720,Psicología y
Pedagogía,3,0.29078415036201477
520746843.1656078461,1656078462,Infantil,1,0.3305356800556183
520746843.1656078461,1656078462,Infantil,2,0.3430086672306061
520746843.1656078461,1656078462,Infantil,3,0.3430086672306061
520746843.1656078461,1656078462,Infantil,4,0.3430086672306061
520746843.1656078461,1656078462,Infantil,5,0.3430086672306061
152397557.1655519553,1655519553,Ciencias Políticas y
Sociales,1,0.3305356800556183
152397557.1655519553,1655519553,Ciencias Políticas y
Sociales,2,0.3430086672306061
365713615.1647644740,1654784718,Cómics,6,0.29078415036201477
365713615.1647644740,1654784718,Cómics,7,0.29078415036201477
365713615.1647644740,1654784718,Cómics,8,0.29078415036201477
365713615.1647644740,1654784718,Cómics,9,0.29078415036201477
1765833037.1655846133,1655846133,Historia,2,0.3305356800556183
1765833037.1655846133,1655846133,Historia,3,0.3305356800556183
1765833037.1655846133,1655846133,Historia,4,0.3430086672306061
265723302.1624443340,1654774564,Libro antiguo y de
ocasion,1,0.3430086672306061
265723302.1624443340,1654774564,Libro antiguo y de
ocasion,2,0.3430086672306061
1504647259.1637074619,1654809482,Novela negra,3,0.3018597960472107
1504647259.1637074619,1654809482,Novela negra,4,0.3430086672306061
1504647259.1637074619,1654809482,Novela negra,5,0.35126230120658875
1504647259.1637074619,1654809482,Novela negra,6,0.35126230120658875
1504647259.1637074619,1654809482,Novela negra,7,0.46425339579582214
1504647259.1637074619,1654809482,Novela negra,8,0.46425339579582214
1504647259.1637074619,1654809482,Novela negra,9,0.46425339579582214
1504647259.1637074619,1654809482,Novela negra,10,0.49770113825798035
1504647259.1637074619,1654809482,Novela negra,11,0.49770113825798035
1504647259.1637074619,1654809482,Novela negra,12,0.49770113825798035
1504647259.1637074619,1654809482,Novela
contemporánea,13,0.3305356800556183
```

Figura 39- Resultados de la ejecución del modelo de predicción de compra.

7.1.5 Petición SQL: obtención transacciones

En la Figura 40 se muestra la petición SQL que se ha generado para obtener los valores de las transacciones reales de los usuarios a partir de los resultados obtenidos en el modelo predictivo.

```
WITH table1 AS(
  SELECT

        hits_product.v2ProductName           AS producto,
        hits_product.v2ProductCategory       AS categoria_producto,
        visitorId                            AS visitorId,
        clientId                             AS clientId,
        h.hitNumber                          AS hitNumber,
        sessions.totals.transactions         AS transaction,
```

```
    FROM `bq-luce.archiever.ga_sessions` AS sessions
    LEFT JOIN UNNEST(hits) AS h
    LEFT JOIN UNNEST(h.product) AS hits_product

),
table2 AS(
  SELECT * FROM `bq-luce.archiever.datos_csv`
)


SELECT table2.client_id, table2.visit_id, table2.categoria_producto,
table2.hit_number,
table1.transaction FROM table1 RIGHT JOIN table2 ON
(table1.visitorId=table2.visit_id
AND table1.clientId=table2.client_id AND
table1.categoria_producto=table2.categoria_producto
AND table1.hitNumber=table2.hit_number)
```

Figura 40- Petición SQL para la obtención de las transacciones reales de los usuarios.