



PROYECTO DE GRADO

Presentado ante la ilustre Universidad de Los Andes como requisito final para
obtener el Título de Ingeniero de Sistemas

DESARROLLO DE UN SISTEMA PARA INFERIR LA DESERCIÓN DE LOS CLIENTES
EN UN E-COMMERCE

BR. DIMITRIO MANDAMADIOTIS KALFAGIANNIS

TUTORES:

YANETH MORENO, M.Sc

MÉRIDA, FEBRERO DE 2023

Índice general

1. Introducción	4
1.1. Antecedentes	5
1.2. Planteamiento del problema	6
1.3. Justificación	6
1.4. Objetivos	7
1.5. Metodología	8
1.6. Alcances	9
2. Marco Referencial	10
2.1. E-Commerce	10
2.2. Modelo	12
2.2.1. Modelo Estadístico	12
2.2.2. Modelos Buy-Till-You-Die (BTYD)	12
2.2.3. Modelo Pareto / NBD	14
2.2.4. Modelo BG / NBD	15
2.3. Data	16
2.3.1. Dataset	17
2.4. Herramientas Tecnológicas	17
2.4.1. Python	17
2.4.2. Biblioteca Lifetimes	17
2.4.3. Biblioteca Pandas	18
2.4.4. Biblioteca Matplotlib	18
2.4.5. Jupyter Notebook	18
2.4.6. Flask	18

2.4.7. Javascript	18
2.4.8. Typescript	19
2.4.9. Vue.js	19
2.4.10. Nuxt	19
2.4.11. Chart.js	20
3. Análisis, Planificación y Diseño del Sistema	21
3.1. Análisis y Definición del Modelo	21
3.2. Arquitectura del Sistema	22
3.3. Requerimientos	24
3.3.1. Modelo	24
3.3.2. API	25
3.3.3. Aplicación Web	26
3.4. Diseño de Interfaz de Usuario	26
4. Desarrollo e Implementación	29
4.1. Implementación de Módulos	29
4.1.1. Modelo	29
4.1.2. API	33
4.1.3. Aplicación Web	35
5. Pruebas y Resultados	42
5.1. Pruebas de Módulos	42
5.1.1. Modelo	42
5.1.2. API	46
5.1.3. Aplicación Web	47
5.2. Resultados	48
5.2.1. Modelo	48
5.2.2. API y Aplicación Web	50
6. Conclusiones y Recomendaciones	52
Bibliografía	54

Capítulo 1

Introducción

El comercio electrónico o también conocido como e-commerce, consiste en la compra y venta de productos o servicios a través de internet, este comenzó a principios de 1970 y se desarrolló a mediados de la década de los 90 como una manera bastante sencilla de adquirir productos o servicios sin tener que desplazarse a una ubicación física.

La cantidad de comercio llevada a cabo electrónicamente ha crecido de manera extraordinaria y esto ha estimulado la creación y utilización de innovaciones como el marketing en internet, cadenas de suministros complejas, sistemas automáticos de recolección de datos y sistemas para la gestión de la relación con el cliente, estos últimos han permitido la aplicación de métodos cuantitativos para el análisis y la optimización del rendimiento en los e-commerce.

Una de las áreas más importantes de estudio es el comportamiento de los clientes, en ella se evalúa y se estudia la relación directa que existe entre el servicio que se les proporciona a los usuarios y los ingresos de la tienda de comercio electrónico, para esto se utilizan variables como el valor del ciclo de vida de un cliente y la tasa de churn y lo que se busca es obtener un incremento en las ventas mediante mejoras en los servicios, la experiencia del cliente y la calidad de los productos.

1.1. Antecedentes

En el mundo de los negocios y el comercio actual seguimos observando un crecimiento rápido de los sistemas digitales y las tecnologías de información, algunos de estos sistemas se basan en la gestión de relaciones con los clientes, los cuales son comunes en negocios contractuales como los servicios de telecomunicaciones o servicios de software basado en suscripción. El comercio electrónico es un negocio no contractual y desde la pandemia, el comercio electrónico se ha convertido en la opción preferida para cualquier tipo de compra (Paraschiv et al., 2022), también en el año 2020 los ingresos en e-commerce aumentaron en un 10 % en Europa (Jílková y Králová, 2021), estas cifras nos demuestran lo vital que es para los negocios mantener a los clientes.

Durante los últimos tres años se han realizado muchos estudios sobre el comportamiento de los clientes en los negocios. (Falla, 2021) en su trabajo de grado Predicción de Abandono de Clientes en Telecomunicaciones Mediante el Aprendizaje Automático, utiliza técnicas y algoritmos de inteligencia artificial para predecir el abandono de clientes en un negocio con modelo contractual. También identificó varias técnicas de minería de datos para la identificación de clientes que están a punto de abandonar.

(Sinha y Raizada, 2022) en su trabajo Modelling Customer Churn Rate and Its Use for Customer Retention Planning muestran un modelo para un negocio contractual basado en máquinas de aprendizaje o machine learning para predecir la tasa de abandono, adicionalmente dan una serie de recomendaciones a los negocios para mantener una tasa de abandono baja según sus descubrimientos.

La complejidad de modelar la tasa de abandono en negocios no contractuales viene de que no se conoce el momento en el que un cliente abandonó. Una manera de aproximar esta variable es mediante el valor de la compra media de un cliente. (Abdolvand et al., 2021) mencionan en su artículo Customer Lifetime Value: Literature scoping map, and an agenda for future research que la literatura se enfoca en explicaciones teóricas para calcular el valor de la compra media de un cliente, pero no expresan las diferencias que pueden existir a la hora de aplicar el conocimiento a la práctica.

Los clientes son el pilar de las tiendas de comercio electrónico y conociendo su probabilidad de desertar en el futuro, los negocios pueden crear herramientas y pro-

cesos para evitar perderlos. (Fu, 2022) propone la construcción de un modelo de alerta temprana, para así reducir el riesgo de abandono de clientes en la industria e-commerce en China. (Seo et al., 2022) presenta un método para incrementar la tasa de compra y prevenir el churn, para ello plantea la emisión de cupones personalizados a los clientes con mayor probabilidad de desertar.

1.2. Planteamiento del problema

En un negocio como una tienda de comercio electrónico lo más importa son los clientes, sin ellos no hay ingresos y sin ingresos la tienda no podría existir. En la actualidad el costo de atraer y convertir personas en nuevos clientes suele ser muy alto, además, es mucho más fácil vender productos a un individuo que ya ha comprado antes en la tienda que a un desconocido, es por esto que vale la pena determinar cuándo los clientes podrían desertar para poder tomar acción antes y tratar de prevenir el abandono. Por otro lado, es esencial determinar con precisión cuántos de los clientes aún siguen siéndolo, con esta información se podría analizar el estado actual del negocio y proyectar escenarios para el futuro.

Calcular el churn de los clientes es más fácil en negocios contractuales, como proveedores de telefonía o de banda ancha, ya que es fácil determinar cuándo estos están a punto de abandonar a medida que sus contratos se acercan a la fecha de finalización. Sin embargo, predecir la deserción en mercados no contractuales como el comercio electrónico es mucho más difícil porque no se conoce el momento de salida de un cliente, y en su lugar debe predecirse.

En este sentido, se considera que desarrollar un sistema capaz de predecir el churn de los clientes en un e-commerce sería de vital importancia, ya que permitiría analizar la salud de un negocio y le daría una ventaja a los administradores y encargados de área de marketing.

1.3. Justificación

El comercio electrónico a nivel mundial ha tenido un crecimiento exponencial en los últimos años y debido a esto se ha generado una cantidad gigantesca de datos

que se pueden utilizar para optimizar métricas y alcanzar objetivos de negocio.

En los negocios contractuales es fácil establecer una base para analizar el comportamiento de los usuarios, esto porque se conoce con exactitud la cantidad de usuarios que dejaron de ser clientes en un momento dado, ya que no renovaron sus contratos o suscripciones. Estos datos permiten a los negocios contractuales crear sistemas complejos de predicciones de rendimiento y la optimización de la experiencia del usuario.

Dicho esto, hay otro grupo de negocios donde no se sabe con exactitud cuando un usuario deja de ser cliente. Es por ello que es necesario definir un sistema para predecir la deserción de los clientes en los negocios no contractuales como las tiendas de comercio electrónico. Este sistema le da la posibilidad a estos negocios de realizar análisis complejos e identificar problemas en las relaciones con sus clientes y tomar acción para prevenir el abandono a tiempo.

1.4. Objetivos

El objetivo general de este trabajo de grado es desarrollar un sistema capaz de inferir la deserción de los clientes en una tienda de comercio electrónico.

Para ello se debe cumplir con los siguientes objetivos específicos:

1. Construir un data Warehouse con información sobre patrones de comportamiento de los clientes en una tienda de comercio electrónico.
2. Definir el método estadístico apropiado para inferir la deserción de los clientes.
3. Desarrollar el modelo para inferir la deserción de los clientes en una tienda de comercio electrónico.
4. Validar el modelo desarrollado con un conjunto de datos de prueba.
5. Evaluar el rendimiento del modelo.

1.5. Metodología

En la actualidad, la agilidad al cambio es un factor de suma importancia en los proyectos que involucren software. Los requisitos y diseños tienden a cambiar rápidamente con el tiempo para adaptarse a las necesidades del proyecto. Por esta razón, existen las metodologías ágiles, estas permiten darle un mayor enfoque al proceso de desarrollo, enfatizar la comunicación cara a cara en lugar de la documentación y en especial facilitar la refactorización y la adaptación a los cambios. En este proyecto se plantea aplicar el método Kanban combinado con un backlog, esto facilita la organización y optimiza el flujo del trabajo para la investigación y el desarrollo del sistema.

Kanban se basa en una estructura de flujo de trabajo continuo de forma visual. Los elementos de trabajo que son representados por tarjetas se organizan en un tablero de kanban, donde pasan de una etapa del flujo de trabajo (columna) a la siguiente. Las etapas habituales del flujo de trabajo son:

- **Por hacer:** que representa el trabajo que no se ha empezado.
- **En curso:** trabajo en el que se está trabajando activamente.
- **En revisión:** trabajo que está finalizado y en espera de revisión.
- **Finalizado:** trabajo completamente terminado.

Una característica importante de kanban es que se establece una cantidad máxima de trabajo que puede existir en cada estado del flujo de trabajo. Limitar la cantidad de trabajo en curso mejora el rendimiento y reducen la cantidad de trabajo “prácticamente listo”, ya que obliga al equipo a centrarse en un conjunto de tareas más pequeño, en este proyecto no se podrá trabajar en más de tres tareas activas. Esta característica puede ser una limitante a la hora de programar las tareas por hacer, es por ello que se va a tratar esta columna como un backlog.

El backlog es una lista de trabajo ordenado por prioridades para el equipo de desarrollo que se obtiene de la hoja de ruta y los requisitos. Los elementos más importantes se muestran al principio del backlog para que el equipo sepa qué hay que entregar primero. En este proyecto el backlog será atendido por el tutor y el autor.

1.6. Alcances

En el presente proyecto se plantea la creación de un sistema que analice los datos relacionados al comportamiento de los usuarios en un sitio web de comercio electrónico, específicamente el historial de transacciones de los usuarios en la tienda. Estos datos servirán de entrada a un modelo que tendrá como salida la probabilidad de que el usuario abandonara la tienda en un momento en el futuro.

El sistema tendrá una interfaz web donde un usuario podrá ingresar a través de un formulario el historial de transacciones de sus clientes, luego el sistema procesará la información y mostrará un gráfico por cada uno de los clientes, este gráfico mostrará la probabilidad de deserción del cliente desde su primera compra hasta 500 días en el futuro.

Para la programación del sistema se hará uso del lenguaje Python para la implementación del modelo ya que es el lenguaje mas utilizado en el área de análisis de datos y la mayoría de las herramientas están codificadas en este lenguaje. Para desarrollar la interfaz web se utilizará el lenguaje Javascript, el framework VueJS y el metaframework Nuxt para el frontend. Para el backend se usará Python y la biblioteca Flask.

Capítulo 2

Marco Referencial

En éste capítulo, se pretende explicar las bases teóricas necesarias para la comprensión del diseño e implementación de un sistema para inferir la deserción de clientes en un e-commerce.

2.1. E-Commerce

Se puede definir al comercio electrónico o E-Commerce como transacciones comerciales realizadas digitalmente entre individuos y organizaciones. Las transacciones comerciales implican un intercambio de valor (por lo general dinero) o comercio entre las partes interesadas a cambio de productos o servicios. El hecho de que se realicen las transacciones digitalmente supone el uso de tecnologías digitales como la internet, la web, dispositivos electrónicos y software.

Cliente

Un cliente en el contexto del comercio electrónico es un individuo que realiza compras regulares a través de algún canal digital de una organización. El proceso de compra de este cliente es similar al de una tienda física y consiste en: visualizar los productos o servicios que ofrece la organización, luego procede a agregar lo que desea comprar a un carrito de compras, cuando el cliente está listo para pagar, el carrito es verificado y luego pagado de forma digital, finalmente se genera un registro

de pedido en el sistema el cual es procesado por la organización para satisfacer la compra.

Pedido

Un pedido u orden de compra es un registro que contiene información relevante sobre una transacción de compra realizada por un cliente en una plataforma de comercio electrónico de una organización. Este registro suele contener información relevante a los procesos de cada organización pero los datos indispensables que incluye son: la fecha en que se realizó un pedido, el valor monetario del mismo y el cliente que lo generó.

Negocio Contractual

Se considera negocio contractual a toda organización que interactúe y trabaje con sus clientes bajo el modelo de suscripción u otra configuración contractual. En este tipo de organizaciones es relativamente fácil determinar si un cliente está activo o no, ya que por lo general se registran los datos de vencimiento y renovación de contratos o suscripciones. Algunos ejemplos de negocios contractuales son las compañías de seguros, empresas de telecomunicaciones, gimnasios y plataformas SaaS.

Negocio No Contractual

Un negocio no contractual es cuando la organización no tiene mecanismo que le ayude a determinar cuando la relación con un individuo o cliente ha finalizado, esto debido a que un cliente puede estar en un periodo de espera antes de realizar otra compra o puede que no compre de nuevo a la organización. Ejemplos de este tipo de negocio son las tiendas de comercio electrónico y las tiendas de venta al por menor.

Deserción de Clientes (Churn)

Deserción de clientes o Churn es la pérdida de clientes debido al desgaste y es lo opuesto a la retención de clientes. Ésta se considera una métrica crucial para las empresas y se utiliza como base para calcular otras métricas de gran importancia

como la vida útil del cliente, el valor de vida útil (CLV por sus siglas en inglés), el gasto de adquisición de clientes y el gasto de éxito del cliente.

Valor de vida útil de un cliente (CLV)

Esta medida representa el valor que tiene un cliente para un negocio a lo largo de toda la relación, es decir, es el valor total que ha aportado un cliente al negocio desde que realizó una primera compra, hasta que desertó.

2.2. Modelo

En líneas generales, un modelo es una representación de un sistema, éste se compone por conceptos que ayudan a explicar el sistema, estudiar los efectos de diferentes componentes y hacer predicciones sobre el comportamiento del mismo. Estos conceptos suelen manifestarse en un grupo de ecuaciones matemáticas.

2.2.1. Modelo Estadístico

Un modelo estadístico es un modelo matemático que representa una o varias suposiciones estadísticas, estas suposiciones tienen describen propiedades que nos permiten calcular la probabilidad de que ocurra un evento.

2.2.2. Modelos Buy-Till-You-Die (BTYD)

Son una familia de modelos estadísticos que tienen como objetivo describir el comportamiento de compra de los clientes como una serie de distribuciones. Estos modelos se basan en dos suposiciones básicas:

1. Los clientes interactúan con la organización cuando están activos o "vivos".
2. Los clientes pueden "morir" o transformarse en clientes inactivos después de un periodo de tiempo.

Existen varios modelos BTYD, pero todos tienen las siguientes características:

Modelado Probabilístico

Todos estos modelos generan distribuciones de probabilidad para describir el comportamiento de los clientes. Si bien los modelos difieren en cómo crean estas distribuciones o cómo se ven estas distribuciones, todos comparten este objetivo principal.

Tabla de pedidos

La entrada requerida para generar las predicciones en todos los modelos BTYD es simplemente una tabla que contenga los pedidos de los clientes. Cada modelo procesa esta tabla de manera diferente, pero todos comparten una estructura de entrada común. Debido a este requisito de entrada, cuantos más datos longitudinales se tenga sobre los pedidos de los clientes, más sólidas serán las predicciones del modelo.

Análisis RFM

El modelado y análisis de RFM es una estrategia común empleada actualmente por las empresas para capturar el comportamiento del cliente y clasificarlo según tres variables:

- **Recencia (Recency):** ¿Hace cuánto tiempo compró un cliente? Este dato hace referencia a la duración en periodos de tiempo entre la primera compra del cliente y la última.
- **Frecuencia (Frequency):** ¿Con qué frecuencia/consistencia compra un cliente? Este valor representa la cantidad de periodos de tiempo en los que el cliente realizó un pedido.
- **Valor Monetario (Monetary):** ¿Cuánto gasta un cliente en promedio? Esto es la suma de todos los valores monetarios de las compras de un cliente dividida por el número total de compras.

Cada uno de estos elementos del análisis RFM juega un papel clave en las distribuciones de los modelos BTYD.

2.2.3. Modelo Pareto / NBD

El nombre se refiere a la distribución de Pareto utilizada para modelar la deserción de clientes y la distribución binomial negativa (NBD) utilizada en la predicción de compras futuras. Este modelos asume que los clientes compran a un ritmo constante durante un periodo de tiempo y luego pasan a la inactividad.

El objetivo del modelo Pareto / NBD es desarrollar el análisis RFM capturando el contexto del cliente al decidir si, el cliente ha desertado o no y con qué frecuencia comprará. Por ejemplo, si estudiamos un cliente A y un cliente B:

- El cliente A compró en una tienda de comercio electrónico cada dos días durante 2 meses. Sin embargo, no ha comprado en las últimas 2 semanas.
- El cliente B compró en la tienda aproximadamente una vez al mes durante los últimos 6 meses. Tampoco ha comprado en las últimas 2 semanas.

A pesar de que ambos clientes tienen la misma antigüedad (14 días), la diferencia en el comportamiento de compra cambia la probabilidad de deserción. Además, a pesar de comprar en la misma tienda, exhiben patrones de compra significativamente diferentes (cada dos días frente a cada mes), por lo que cualquier predicción debe tener en cuenta sus hábitos de compra.

El modelo de Pareto/NBD se basa en cinco supuestos:

Desde el punto de vista del cliente individual:

- **Compras siguen la distribución Poisson:** Mientras está activo, el número de transacciones realizadas por un cliente en un período de tiempo de longitud t se representan mediante una distribución Poisson con media λt . En realidad, Pareto/NBD, como su nombre lo indica, utiliza una distribución binomial negativa de la cual, la distribución de Poisson es un caso especial. La distribución de Poisson es una elección acertada para modelar el tiempo entre transacciones, ya que diferentes negocios tendrán diferentes tiempos y los clientes también pueden exhibir diferentes patrones de compra.
- **“Vida útil” exponencial:** Cada cliente tiene una “vida útil” no observada de longitud τ . Este punto en el que el cliente se vuelve inactivo se distribuye

exponencialmente con una tasa de deserción μ . Esto lo que nos dice es que en general, la mayoría de los clientes se transformarán en inactivos relativamente rápido después de realizar una compra inicial pero una pequeña cohorte de clientes permanecerá durante mucho tiempo.

Heterogeneidad entre clientes:

- **Las tasas de compra individuales siguen la distribución gamma:** la tasa de compra λ para los diferentes clientes sigue una distribución gamma entre la población de clientes.
- **Tasa de deserción sigue distribuciones gamma diferentes:** Las tasas de deserción μ siguen una distribución gamma que es diferente entre los clientes.
- **Independencia de tasas:** Las tasas de transacciones λ y las tasas de deserciones μ están distribuidas independientemente entre ellas.

Este modelo tiene mucho poder a la hora de analizar el comportamiento de clientes y requiere solo dos entradas sobre cada cliente, su recenciaz "frecuencia". El modelo utiliza la inferencia de probabilidad logarítmica para hacer predicciones sobre el futuro. Un problema de este modelo es que su implementación puede ser desafiante desde el punto de vista computacional debido al uso prolongado de la función hipergeométrica gaussiana.

2.2.4. Modelo BG / NBD

Este modelo deriva del modelo Pareto / NBD como una alternativa simple de calcular desde el punto de vista computacional. Su nombre hace referencia a la distribución geométrica beta utilizada para modelar la deserción de clientes y la distribución binomial negativa (NBD) utilizada en la predicción de compras futuras.

La mayoría de los aspectos del modelo BG / NBD reflejan directamente los del modelo Pareto / NBD. La única diferencia radica en las consideraciones que definen cómo y cuándo los clientes se vuelven inactivos. La distribución de tiempo de Pareto asume que la deserción puede ocurrir en cualquier momento, independientemente

de la ocurrencia de compras reales, en este modelo se asume que el abandono ocurre inmediatamente después de una compra, permitiendo así modelar este proceso mediante la distribución geométrica beta (BG por sus siglas en inglés).

El modelo de BG / NBD se basa en cinco supuestos :

1. Mientras está activo, el número de transacciones realizadas por un cliente representan mediante una distribución Poisson con una tasa de transacciones λ . Esto es equivalente a suponer que el tiempo entre transacciones se distribuye exponencialmente con tasa de transacción λ .
2. La tasa de compra λ para los diferentes clientes sigue una distribución gamma entre la población de clientes.
3. Después de cualquier transacción, un cliente se vuelve inactivo con probabilidad p . Por lo tanto, el punto en el que el cliente "abandona" se distribuye entre las transacciones de acuerdo con una distribución geométrica (desplazada).
4. Las tasas de transacciones λ y las tasas de deserciones μ están distribuidas independientemente entre ellas.
5. La heterogeneidad en p sigue una distribución beta.

El modelo BG / NBD se puede implementar de una manera relativamente sencilla y eficiente, además de que la estimación de sus parámetros no requiere ningún software especializado ni la evaluación de funciones matemáticas no convencionales.

2.3. Data

Datos o data en inglés, es una representación simbólica de un atributo o variable cuantitativa o cualitativa. Los datos describen hechos empíricos, sucesos y entidades, es decir, transmiten información.

Los datos son indispensables en la investigación científica, las finanzas y en prácticamente cualquier otra forma de actividad organizacional humana. Ejemplos de conjuntos de datos incluyen precios de acciones, tasas de delincuencia, pedidos de clientes en un e-commerce, tasas de desempleo, tasas de alfabetización y datos del censo.

2.3.1. Dataset

Un Dataset se puede definir como un conjunto de información en particular, representado en una tabla o matriz de análisis. La tabla está conformada por columnas y cada una de ellas representa una variable de datos, las filas suponen un grupo de datos específicos.

2.4. Herramientas Tecnológicas

Datos o data en inglés, es una representación simbólica de un atributo o variable cuantitativa o cualitativa. Los datos describen hechos empíricos, sucesos y entidades, es decir, transmiten información.

Los datos son indispensables en la investigación científica, las finanzas y en prácticamente cualquier otra forma de actividad organizacional humana. Ejemplos de conjuntos de datos incluyen precios de acciones, tasas de delincuencia, pedidos de clientes en un e-commerce, tasas de desempleo, tasas de alfabetización y datos del censo.

2.4.1. Python

Python es un lenguaje de programación dinámico de propósito general extremadamente popular. Éste lenguaje soporta múltiples paradigmas de programación como lo son la programación orientada a objetos, la programación funcional y la programación estructurada. Su filosofía hace énfasis en la legibilidad del código mediante el uso de indentación y se considera una de las herramientas principales en área de las ciencias de datos.

2.4.2. Biblioteca Lifetimes

Lifetimes es una biblioteca de software que implementa los modelos estadísticos Buy-Till-You-Die, Customer Lifetime Value y el modelo bayesiano PyMC (en Beta) en el lenguaje de programación Python. Es de código abierto bajo la licencia Apache 2.0.

2.4.3. Biblioteca Pandas

Pandas es una biblioteca de software escrita para el lenguaje de programación Python, comúnmente usada para la manipulación y el análisis de datos. Ofrece estructuras de datos y operaciones para manipular datos de forma tabular y series de tiempo. Es software libre publicado bajo la licencia BSD de tres cláusulas.

2.4.4. Biblioteca Matplotlib

Matplotlib es una biblioteca de gráficos para el lenguaje de programación Python. Se utiliza para incrustar gráficos en aplicaciones y poder visualizar los datos o resultados de modelos. Es de código abierto y se distribuye bajo la licencia de estilo BSD.

2.4.5. Jupyter Notebook

Jupyter Notebook (anteriormente IPython Notebook) es un entorno computacional interactivo muy popular en la ciencia de datos, el cual se ejecuta en la web y se utiliza para crear documentos que simulan un cuaderno digital. Cada documento contiene una lista ordenada de celdas de entrada/salida que pueden contener código de programación, texto (usando el formato Markdown), lenguaje matemático, diagramas e imágenes.

2.4.6. Flask

Flask es un framework de Python que se utiliza para crear aplicaciones y sistemas web. Este framework no requiere otro tipo de herramientas y bibliotecas, tampoco tiene componentes preexistentes lo que lo hace muy ligero. Es de código abierto bajo la licencia BSD y fue lanzado inicialmente en abril del año 2010.

2.4.7. Javascript

Javascript es un lenguaje de programación dinámico de alto nivel, es interpretado y su sintaxis se basa en los lenguajes C++ y Java. Todos los navegadores web modernos interpretan el código Javascript por lo que se considera el lenguaje de la

web, se utiliza principalmente para añadir dinamismo a las páginas web en el lado del cliente, pero actualmente se utiliza también en el lado del servidor para crear aplicaciones de todo tipo.

2.4.8. Typescript

Typescript es un lenguaje de programación tipado, compilado y de código abierto bajo la licencia Apache. Fue desarrollado en Octubre del año 2012 y actualmente es mantenido por Microsoft como un subconjunto del lenguaje de programación Javascript.

El uso y popularidad de Typescript ha crecido exponencialmente en los últimos años debido a las mejoras que ofrece sobre el lenguaje Javascript, en especial la validación de tipos para las variables lo cual ayuda en la detección temprana de errores aumentando la robustez del software, se considera esencial para proyectos grandes.

2.4.9. Vue.js

Vue.js es un framework de código abierto escrito en Typescript que se utiliza en la construcción de interfaces de usuario dinámicas y aplicaciones web. Fue creado por Evan You en febrero del año 2014 bajo la licencia MIT. Este framework es de los más populares actualmente para crear interfaces web y se enfoca en una arquitectura de representación declarativa y composición de componentes.

2.4.10. Nuxt

Nuxt es un metaframework que utiliza como base a Vue.js, es de código abierto bajo la licencia MIT y tiene como objetivo facilitar la creación de aplicaciones web “universales”, es decir, sitios web que tengan la funcionalidad de una aplicación web y los beneficios de los sitios web estáticos. Fue creado en octubre del 2016 por Sebastien Chopin, Alexandre Chopin y Pooya Parsa.

2.4.11. Chart.js

Es una biblioteca gratuita de código abierto para la visualización de datos, está escrita en Javascript y fue creada en el año 2013 bajo la licencia MIT. Esta biblioteca permite mostrar datos en gráficos de barra, línea, área, circular, burbujas, radar, polar y de dispersión, todo esto utilizando la tecnología de HTML5 Canvas. Es una de las bibliotecas mas populares y se considera la mejor por su simplicidad de uso.

Capítulo 3

Análisis, Planificación y Diseño del Sistema

En este capítulo se explican los procesos que fueron necesarios para la investigación y creación del sistema para inferir la deserción de clientes. Esta es la fase inicial donde se analizan los objetivos del proyecto y se definen las líneas de diseño base para la creación del sistema.

3.1. Análisis y Definición del Modelo

Como se describió anteriormente, el ámbito no contractual y continuo en el que se desarrolla el comercio electrónico, es el más común pero el más desafiante a la hora de evaluar, analizar y obtener información de valor para los negocios.

Inicialmente se evaluó la posibilidad de utilizar técnicas de inteligencia artificial o aprendizaje automático, pero debido al escenario en el que se está trabajando, donde la deserción del cliente no es explícitamente observable y puede ocurrir en cualquier momento, se dificulta diferenciar entre los clientes que se han ido indefinidamente y los que volverán en el futuro, además, la naturaleza y escasez de datos de acceso público, no permite entrenar un modelo complejo que pudiese predecir ese comportamiento.

Dicho esto, se buscó una solución mediante uno o varios modelos estadísticos. Este tipo de enfoque es considerado efectivo para este ámbito, ya que se pueden

generar ecuaciones predeterminadas que se adapten a los datos obtenidos, inferir la relación que existe entre las variables y así generar valores esperados.

Luego de decidir el enfoque de la solución, se comenzó una investigación exhaustiva para determinar que variables modelar y que relaciones evaluar. Este problema ya se ha atacado antes, en el artículo Counting Your Customers: Who Are They and What Will They Do Next? (Schmittlein et al., 1987) se plantea una solución que se basa en predecir el numero de transacciones de un cliente en el futuro, si bien los resultados de este modelo son efectivos, existe un problema de implementación desde el punto de vista computacional debido a la complejidad de las funciones matemáticas utilizadas en el modelo.

En el artículo “Counting Your Customers” the Easy Way: An Alternative to the Pareto/NBD Model (Fader et al., 2005), los autores realizaron una mejora considerable al modelo Pareto / NBD, simplificando así su implementación y manteniendo los resultados.

Posteriormente, en una nota A Step-by-Step Derivation of the BG/NBD Model (Peter S. Fader, Bruce G. S. Hardie, Ka Lok Lee 2019) los autores describieron los pasos para derivar los resultados obtenidos anteriormente, lo que permitió la implementación del modelo en código Python por parte de Cameron Davidson-Pilon, creando así la biblioteca Lifetimes que será el pilar base del sistema para predecir la deserción de clientes en un e-commerce.

3.2. Arquitectura del Sistema

La solución al problema planteado consta de un sistema donde se integran tres módulos: el modelo, la API y la aplicación web (Figura). El módulo de modelo se encarga de procesar la data que se obtuvo y generar un modelo estadístico para la predicción de la tasa de deserción de los clientes. Este proceso se realiza una vez, es decir, no es necesario crear otro modelo o ajustar el modelo actual con los datos ingresados por un usuario mediante la interfaz.

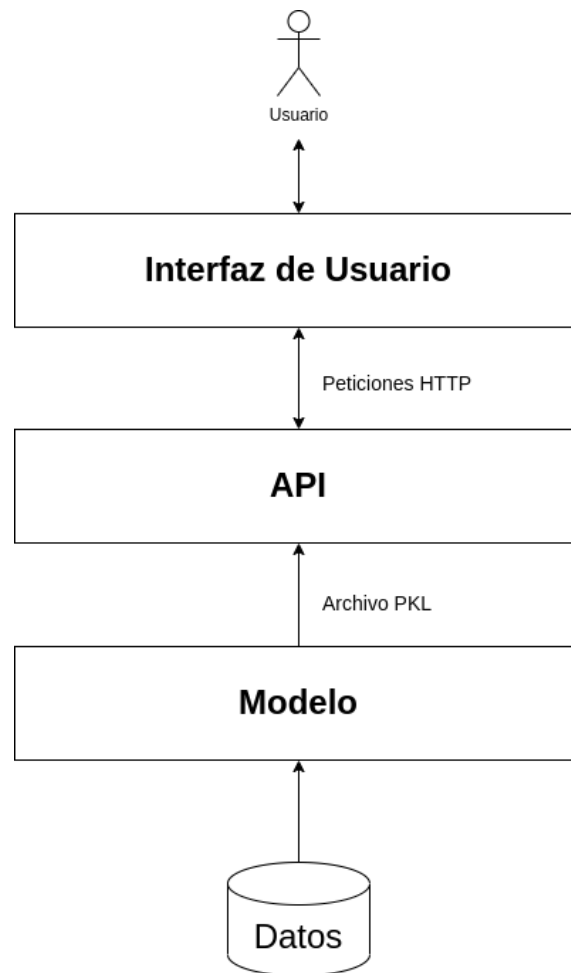


Figura 3.1: Arquitectura del Sistema

La API sirve de middleware o capa intermedia entre el modelo y la interfaz del usuario, es decir, esta se encargaría de recibir los datos de la aplicación web, procesarlos haciendo uso del modelo y devolver el resultado de nuevo a la interfaz para que el usuario pueda verlos.

Finalmente, la interfaz tiene como finalidad permitir introducir al usuario al sistema, recibir su entrada como una lista de transacciones que están representadas por una fecha y un identificador de cliente, enviar esa información a la API para ser procesada, recibir el resultado y mostrarle un gráfico de probabilidad por cada cliente donde se exprese la probabilidad de que este haya desertado en un punto en el tiempo futuro.

La comunicación entre el modulo del modelo y la API se realizaría mediante un archivo PKL que contiene los parámetros del modelo creado, el cual será importado por la API al momento de su ejecución y utilizado para generar predicciones. La API y la aplicación web hablarían mediante peticiones HTTP e intercambiarían la información en formato JSON.

3.3. Requerimientos

A pesar de que la metodología de desarrollo es ágil, se consideró importante definir una serie de requerimientos como base que debe cumplir el sistema, esto con la finalidad de mantener un enfoque durante la implementación y que las tareas del backlog tengan un objetivo en común.

Acorde a lo descrito en la sección anterior en la cual se desarrolló una arquitectura para el sistema, la lista de requerimientos por módulo que se estableció es la siguiente:

3.3.1. Modelo

Requerimientos Funcionales

- El sistema genera un modelo estadístico que pueda predecir la deserción de clientes de un e-commerce.

Requerimientos no Funcionales

- Se debe importar las transacciones de un archivo CSV.
- Se debe procesar la data en un dataframe con las características definidas por la biblioteca lifetimes.
- El modelo se debe guardar en un archivo con formato PKL en la carpeta del módulo API.
- El código fuente del modelo debe estar disponible dentro del repositorio del proyecto en GitHub.
- Se debe implementar el modelo en un Jupyter Notebook.

- En el Jupyter Notebook debe estar expresado el proceso de generación del modelo.
- Se deben mostrar ejemplos de uso del modelo.

3.3.2. API

Requerimientos Funcionales

- Crear una ruta (/conditional-probability-alive) donde se reciba una lista de transacciones por usuario y se retorne la probabilidad de vida de cada cliente en 500 unidades de tiempo desde la primera transacción.

Requerimientos no Funcionales

- La API será usada por una aplicación web.
- La API usará la arquitectura REST.
- El formato de entrada y salida debe estar descrito en el archivo README.md correspondiente a la carpeta del proyecto.
- Las instrucciones de ejecución deben estar contempladas en el archivo README.md correspondiente a la carpeta del proyecto.
- El formato debe especificar en cada campo con su clave y tipo de dato aceptado.
- Permitir el acceso libre a la API.
- La API debe procesar y retornar datos en formato JSON.
- La API debe estar disponible de manera pública mediante un enlace de prueba.
- El código fuente de la API debe estar disponible dentro del repositorio del proyecto en GitHub.

3.3.3. Aplicación Web

Requerimientos Funcionales

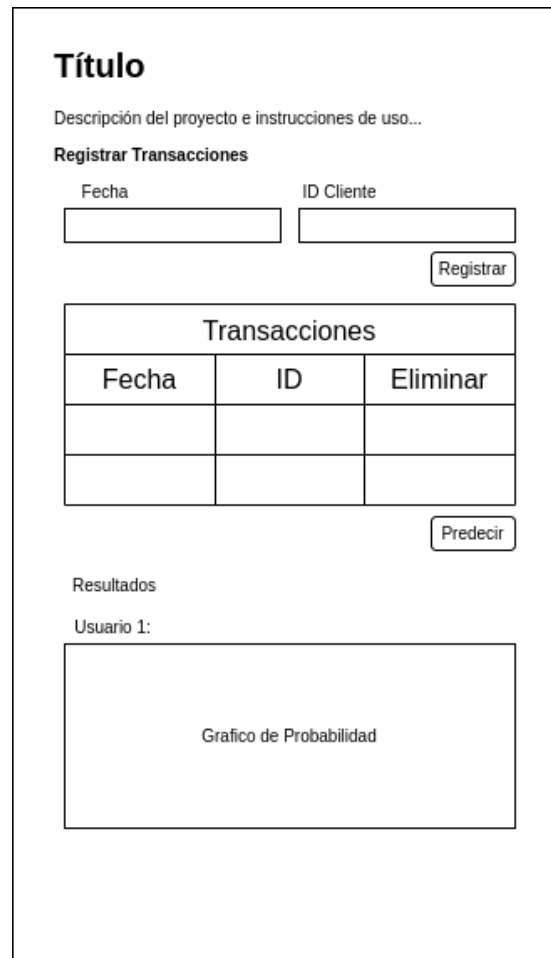
- El usuario ingresa la lista de transacciones de los clientes y obtiene la probabilidad de que los clientes desertara de su negocio en un grafico lineal.

Requerimientos no Funcionales

- Se debe validar que las fechas ingresadas por el usuario sigan el formato DD-MM-AAAA.
- No se pueden ingresar o enviar transacciones sin fecha o identificador de cliente.
- El identificador de cliente es una cadena de caracteres con tamaño mayor de tres.
- Las instrucciones de ejecución deben estar contempladas en el archivo README.md correspondiente a la carpeta del proyecto.
- Se debe describir brevemente las instrucciones de uso en el archivo README.md correspondiente a la carpeta del proyecto.
- La aplicación web debe procesar y retornar datos en formato JSON.
- Se debe describir brevemente las instrucciones de uso la interfaz.
- Se debe acceder a la API mediante un método POST.
- El código fuente de la aplicación web debe estar disponible dentro del repositorio del proyecto en GitHub.

3.4. Diseño de Interfaz de Usuario

Lo más importante en el sistema es permitir a un usuario ingresar una lista de transacciones de clientes y que este luego pueda ver la probabilidad de que estos clientes deserten, es por ello que se plantea una interfaz simple de una sola página web (ver Figura 2).



The mockup shows a web interface with the following components:

- Título**: A section header.
- Descripción del proyecto e instrucciones de uso...**: A text area for project description and instructions.
- Registrar Transacciones**: A section header for the transaction registration form.
- Fecha**: A text input field for the date.
- ID Cliente**: A text input field for the client ID.
- Registrar**: A button to submit the transaction.
- Transacciones**: A table header for the transaction list.
- Table**: A table with 3 columns: Fecha, ID, and Eliminar. It contains two empty rows for data.
- Predecir**: A button to predict the probability of desertion.
- Resultados**: A section header for the results.
- Usuario 1:**: A text label for the user.
- Grafico de Probabilidad**: A placeholder for a probability graph.

Figura 3.2: Intefaz de Usuario

La página web inicialmente muestra información sobre el proyecto como título, descripción e instrucciones sobre como utilizar el sistema, luego se muestra un formulario de entrada donde el usuario puede registrar una transacción. La transacción consiste de una fecha en formato DD-MM-AAAA y un identificador de cliente.

La experiencia de usuario en el sistema es simple, cuando se registra una transacción valida, esta se agrega a una tabla que el usuario puede visualizar con la opción de eliminar alguna transacción ya registrada.

El usuario al terminar de registrar las transacciones hace clic en el botón “predecir”, luego de esto se muestra un gráfico que representa la probabilidad de desertar de un cliente a lo largo del tiempo y comenzando desde la fecha de la primera transac-

ción registrada del cliente. Se construye un grafico por cada identificador de cliente ingresado.

Capítulo 4

Desarrollo e Implementación

En este capítulo se explican en detalle los procesos que fueron necesarios para implementar el sistema tomando en cuenta la arquitectura planteada en el capítulo anterior.

4.1. Implementación de Módulos

A continuación se aprecia la implementación de cada uno de los módulos que componen el sistema para inferir la deserción de los clientes en un e-commerce.

4.1.1. Modelo

El objetivo de este módulo es generar un modelo en el formato PKL, para esto se decidió crear un programa utilizando la tecnología Jupyter Notebook. El proceso se realizó en fases:

Instalación de dependencias

La primera fase consiste en importar las dependencias necesarias para realizar el análisis, preprocesamiento y generar el modelo:

```
1 import pandas as pd
2 import datetime
3 from lifetimes import BetaGeoFitter
4 from lifetimes.plotting import plot_frequency_recency_matrix
```

```
5 from lifetimes.plotting import plot_probability_alive_matrix
6 from lifetimes.plotting import plot_period_transactions
7 from lifetimes.plotting import
    plot_calibration_purchases_vs_holdout_purchases
8 from lifetimes.plotting import plot_history_alive
9 from lifetimes.utils import summary_data_from_transaction_data
10 from lifetimes.utils import calibration_and_holdout_data
```

Listing 4.1: Importar dependencias en modelo.ipynb

Cargar el Dataset

El Dataset que se va a utilizar contiene los datos de transacciones de ventas de una tienda de comercio electrónico con sede en el Reino Unido. Esta tienda de Londres vende regalos y artículos para el hogar y sus clientes provienen de todo el mundo.

El conjunto de datos contiene 500.000 Filas y 8 columnas en las cuales esta representadas las transacciones de ventas de la tienda durante un año. Las columnas son las siguientes:

- **TransactionNo (categórico):** un número único de seis dígitos que define cada transacción. La letra “C” en el código indica una cancelación.
- **Date (numérico):** la fecha en que se generó cada transacción.
- **ProductNo (categórico):** un carácter único de cinco o seis dígitos que se utiliza para identificar un producto específico.
- **Product (categórico):** nombre del producto/artículo.
- **Price (numérico):** el precio de cada producto por unidad en libras esterlinas (£).
- **Quantity (numérico):** la cantidad de cada producto por transacción. Valores negativos relacionados con transacciones canceladas.
- **CustomerNo (categórico):** un número único de cinco dígitos que define a cada cliente.

- **Country (categórico):** nombre del país donde reside el cliente.

Hay un pequeño porcentaje de cancelación de pedidos en el conjunto de datos. La mayoría de estas cancelaciones se debieron a condiciones de falta de existencias en algunos productos. En esta situación, los clientes tienden a cancelar un pedido porque quieren que todos los productos se entreguen de una vez. Los datos se cargan desde el archivo CSV utilizando la biblioteca Pandas.

```
1 transactional_data = pd.read_csv('../data/Sales Transaction v.4a.csv')
   '
```

Listing 4.2: Cargar Dataset en modelo.ipynb

Preprocesamiento de los Datos

Para poder trabajar correctamente con los datos, es necesario que estos cumplan con los formatos esperados de la biblioteca lifetimes y que los datos sean válidos, para ello se realizaron las siguientes operaciones sobre el dataset:

1. Se removió la información de precio, país, producto y cantidad ya que no es necesaria para el análisis.
2. Las transacciones canceladas se van a considerar en el modelo como una interacción de parte del cliente, por lo tanto no se eliminarán.
3. Varios productos pueden pertenecer a una misma transacción, para representar esto, el dataset contiene entradas con el mismo número de transacción pero con productos diferentes, como solo es necesario saber si se realizó una transacción y no importan los detalles de la misma, se eliminarán las entradas duplicadas.
4. Es necesario transformar los datos en el campo de fecha al formato datetime de Python.

El código que realiza este preprocesamiento es el siguiente:

```
1 # Operacion 1
2 transactional_data = transactional_data.drop(columns=['Price', '
   Quantity', 'Country', 'ProductName', 'ProductNo'])
```

```
3
4 # Operacion 3
5 transactional_data = transactional_data.drop_duplicates(subset=[ '
    TransactionNo' ])
6
7 # Operacion 4
8 transactional_data['Date'] = pd.to_datetime(transactional_data['Date
    '], format='%m/%d/%Y')
```

Listing 4.3: Preprocesamiento en modelo.ipynb

Formato de datos RFM

Los modelos BTYD y específicamente la biblioteca lifetimes reciben como entrada datos en formato RFM, esto quiere decir que debemos llevar los datos transaccionales a una tabla RFM.

Para transformar nuestra data transaccional a data RFM utilizamos la función `summary_data_from_transaction_data` de la biblioteca lifetimes.

```
1 rfm_data = summary_data_from_transaction_data(transactional_data, '
    CustomerNo', 'Date', observation_period_end='2019-12-31')
```

Listing 4.4: Formato RFM en modelo.ipynb

Ajustar el modelo

Finalmente se va crear un modelo BG/NBD con la data RFM que representa el comportamiento de los clientes del dataset, para ello se utiliza la clase `BetaGeoFitter` de la biblioteca lifetimes.

```
1 model = BetaGeoFitter()
2 model.fit(rfm_data['frequency'], rfm_data['recency'], rfm_data['T'])
```

Listing 4.5: Ajustar Modelo en modelo.ipynb

Guardar modelo

La clase `BetaGeoFitter` de lifetimes posee una rutina que permite guardar el modelo en un formato PKL para poder ser usado posteriormente.


```
1 model.save_model('../api/modelo.pkl')
```

Listing 4.6: Guardar Modelo en modelo.ipynb

4.1.2. API

Para implementar la API en el lenguaje de programación Python, se utilizó la biblioteca Flask que abstrae toda la funcionalidad relacionada con programar un servidor web. Esta biblioteca se encarga de implementar todas las reglas del protocolo HTTP.

El objetivo de la API es crear una ruta HTTP que exponga el recurso conditional-probability-alive, el cual permite calcular la probabilidad de vida de uno o mas clientes. El código del servidor web es el siguiente:

```
1 # Dependencias
2 import datetime
3 import pandas as pd
4 from flask import Flask, request
5 from flask_cors import CORS
6 from lifetimes import BetaGeoFitter
7 from lifetimes.utils import summary_data_from_transaction_data
8 from lifetimes.utils import calculate_alive_path
9
10 # Cargar Modelo
11 model = BetaGeoFitter()
12 model.load_model("./modelo.pkl")
13
14 app = Flask(__name__)
15 CORS(app)
16
17 # Ruta para verificar la salud del servidor
18 @app.route("/")
19 def index():
20     return "Index Page"
21
22 # Recurso
23 @app.route("/conditional-probability-alive", methods=["POST"])
24 def conditionalProbabilityAlive():
```

```
25     if request.method == "POST":
26         customer_field = "customer"
27         date_field = "date"
28         data = request.json["transactions"]
29
30         # Validaciones
31         if data.get("customer") is None or data.get("date") is None:
32             return "Bad Request", 400
33
34         if type(data.get("customer")) not in (tuple, list):
35             return "Bad Request", 400
36
37         if type(data.get("date")) not in (tuple, list):
38             return "Bad Request", 400
39
40         if len(data.get("date")) != len(data.get("customer")):
41             return "Bad Request", 400
42
43         df = pd.DataFrame.from_dict(data)
44         rfm_data = summary_data_from_transaction_data(df,
45 customer_field, date_field)
46         rfm_data["p_alive"] = model.conditional_probability_alive(
47             rfm_data["frequency"], rfm_data["recency"], rfm_data["T"]
48         )
49         result = rfm_data.to_dict("index")
50
51         for i, _row in rfm_data.iterrows():
52             path = (
53                 calculate_alive_path(
54                     model, df[df[customer_field] == i], date_field,
55 500
56                     )
57                 .to_numpy()
58                 .tolist()
59             )
60             result[i]["path"] = [item for sublist in path for item
61 in sublist]
62         return {"result": result}
```

```
60     else:
61         raise RuntimeError("Unsupported method {}".format(request.
            method))
```

Listing 4.7: main.py

Al inicio, el programa importa las dependencias necesarias para su ejecución: Pandas, datetime, lifetimes y Flask; luego de esto, se inicializa y carga el modelo, acto seguido, se implementa una ruta de inicio para verificar que el servidor se está ejecutando y finalmente se implementa el recurso objetivo.

El recurso `/conditional-probability-alive` recibe una lista de transacciones de clientes, transforma esa información en una tabla RFM que es utilizada como entrada al modelo para calcular el “camino de vida” o probabilidad de que cada cliente esté activo durante 500 días luego de la primera transacción.

El valor de 500 días se puede parametrizar, pero se decidió mantenerlo constante para evitar una sobrecarga del servidor en caso de que se solicite procesar mucha información. Finalmente el recurso retorna toda la información que ha calculado, incluyendo la tabla RFM.

4.1.3. Aplicación Web

Para construir la interfaz web se usó Typescript como lenguaje de programación, Vue.js como framework y Nuxt como metaframework para ahorrar tiempo de desarrollo. Vue.js se encarga de abstraer y gestionar todo el dinamismo, mientras que Nuxt provee de una estructura al proyecto y abstrae toda la lógica de compilación, optimización y manejo de dependencias del proyecto. La aplicación esta compuesta de una sola vista llamada “index” donde se representan todos los aspectos de la interfaz. La vista ejecuta varias funciones según la acción del usuario.

En la programación de la vista se utilizaron variables de estado internas para almacenar la información ingresada por el usuario, realizar cálculos intermedios y guardar la respuesta de la API, estas variables tienen la siguiente definición:

```
1 const runtimeConfig = useRuntimeConfig()
2 const transactions = ref({
3   customer: [] as string[],
4   date: [] as string[]
```

```
5 })
6 const newTransaction = ref({
7   date: '',
8   customer: ''
9 })
10 const formErrors = ref({
11   date: false,
12   client: false
13 })
14 const customerData = ref({} as {
15   [k: string]: {
16     startDate: string,
17     transactions: string[]
18   }
19 })
20 const result = ref({} as {
21   [k: string]: {
22     p_alive: number,
23     dates: string[],
24     path: number[]
25     T: number,
26     recency: number,
27     frequency: number
28   }
29 })
```

Listing 4.8: script en index.vue

Cuando el usuario presiona el botón Registrar” se ejecuta la función `.addTransaction`”, la cual se encarga de validar y registrar una transacción, su implementación es la siguiente:

```
1 function addTransaction () {
2   formErrors.value.date = false
3   formErrors.value.client = false
4
5   const utc = moment(newTransaction.value.date, 'DD-MM-YYYY', true)
6
7   if (!utc.isValid()) {
8     formErrors.value.date = true
```

```
9   }
10
11   if (!newTransaction.value.customer) {
12     formErrors.value.client = true
13   }
14
15   if (formErrors.value.date || formErrors.value.client) {
16     return false
17   }
18
19   transactions.value.customer.push(newTransaction.value.customer)
20   transactions.value.date.push(moment(newTransaction.value.date, 'DD-MM-YYYY').format('DD-MM-YYYY'))
21
22   if (!customerData.value[newTransaction.value.customer]) {
23     customerData.value[newTransaction.value.customer] = {
24       startDate: moment().format('DD-MM-YYYY'),
25       transactions: []
26     }
27   }
28
29   if (moment(customerData.value[newTransaction.value.customer].startDate, 'DD-MM-YYYY') > moment(newTransaction.value.date, 'DD-MM-YYYY')) {
30     customerData.value[newTransaction.value.customer].startDate = moment(newTransaction.value.date, 'DD-MM-YYYY').format('DD-MM-YYYY')
31   }
32
33   customerData.value[newTransaction.value.customer].transactions.push(moment(newTransaction.value.date, 'DD-MM-YYYY').format('DD-MM-YYYY'))
34
35   newTransaction.value = {
36     date: '',
37     customer: ''
38   }
```

```
39 }
```

Listing 4.9: addTransaction en index.vue

Luego de agregar una transacción, se genera una tabla en la interfaz donde se puede ver la información registrada. En cada entrada de la tabla existe un botón para eliminar la transacción que al ser accionado por el usuario ejecuta la función removeTransaction” que tiene la siguiente implementación:

```
1 function removeTransaction (id: number) {  
2   if (id === 0) {  
3     transactions.value.customer.shift()  
4     transactions.value.date.shift()  
5   } else {  
6     transactions.value.customer.splice(id, 1)  
7     transactions.value.date.splice(id, 1)  
8   }  
9 }
```

Listing 4.10: removeTransaction en index.vue

Luego de que el usuario registre las transacciones que desee, acciona el botón ”Predecir.”^{el} cual ejecuta la función ”predictz” tiene la siguiente lógica:

```
1 async function predict () {  
2   const response = await $fetch(`${runtimeConfig.public.apiBase}/  
   conditional-probability-alive`, {  
3     headers: {  
4       'Content-Type': 'application/json'  
5     },  
6     method: 'POST',  
7     body: {  
8       transactions: {  
9         customer: transactions.value.customer,  
10        date: transactions.value.date.map(e => moment(e, 'DD-MM-YYYY'  
11        ).format('MM/DD/YYYY'))  
12      }  
13    }  
14  }) as any  
15  result.value = response.result  
16  for (const customer in result.value) {
```

```
17     const startDate = customerData.value[customer].startDate
18     result.value[customer].dates = generateDates(startDate, 500)
19     result.value[customer].path = result.value[customer].path.map(v
20     => Math.floor((1 - v) * 1000) / 10)
21   }
```

Listing 4.11: predict en index.vue

Existe una función auxiliar que genera las fechas diarias desde la menor fecha en una transacción, hasta 500 días luego, su implementación es la siguiente:

```
1 function generateDates (start: string, end: number) {
2   const dateArray = []
3   const currentDate = moment(start, 'DD-MM-YYYY').toDate()
4   const endDate = new Date(currentDate).setDate(currentDate.getDate
5   () + end)
6
7   while (currentDate <= new Date(endDate)) {
8     dateArray.push(moment(currentDate).format('DD-MM-YYYY'))
9     // Use UTC date to prevent problems with time zones and DST
10    currentDate.setUTCDate(currentDate.getUTCDate() + 1)
11  }
12  return dateArray
13 }
```

Listing 4.12: generateDates en index.vue

El resultado se muestra en gráficos generados mediante la biblioteca Chart.js, para ello se ejecuta el siguiente código en Vue:

```
1   <div v-if="Object.keys(result).length" class="m-auto max-w-5xl">
2     <h2 class="font-bold text-xl pb-6">
3       Resultados
4     </h2>
5     <div v-for="(value, name) in result" :key="name" class="pb-4">
6       <h3 class="font-bold">
7         Usuario: {{ name }}
8       </h3>
9       <LineChart
10         class="w-full"
11         :chart-data="{
```

```
12         labels: value.dates,
13         datasets: [
14             {
15                 data: value.path,
16                 label: '% Probabilidad de haber desertado',
17                 spanGaps: true,
18                 pointRadius: 0,
19                 borderColor: 'rgba(200, 0, 0, 0.8)',
20                 backgroundColor: 'rgba(200, 0, 0, 0.2)',
21             }
22         ]
23     }"
24     :options="{
25         scales: {
26             y: {
27                 ticks: {
28                     callback(value: number) {
29                         return value + ' %';
30                     }
31                 }
32             }
33         }
34     }"
35     />
36 </div>
37 </div>
```

Listing 4.13: crear gráfico en index.vue

Finalmente obtenemos una interfaz completa que cumple con los requerimientos del capítulo anterior y se puede ver en la figura a continuación.

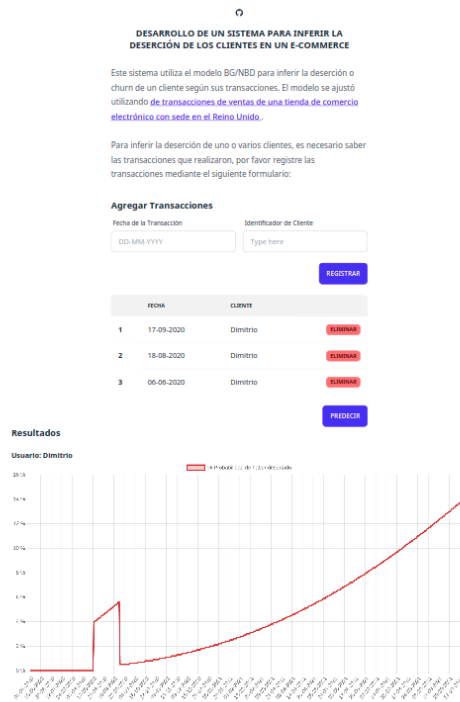


Figura 4.1: Vista Index

Capítulo 5

Pruebas y Resultados

En este capítulo se explican en detalle los resultados y pruebas realizadas para garantizar que el funcionamiento y rendimiento de cada módulo cumpla los objetivos establecidos para el sistema.

5.1. Pruebas de Módulos

A continuación se presentan las pruebas realizadas por cada módulo del sistema.

5.1.1. Modelo

Para validar el modelo se realizaron varias pruebas y análisis de la información obtenida, estas fueron las conclusiones obtenidas.

Matriz de Frecuencia, Recencia y Compras Esperadas

Inicialmente lo que se busca con el modelo es predecir el número esperado de transacciones que un cliente realizará el próximo periodo de tiempo, es por esto que se puede validar la calidad del funcionamiento del modelo graficando la relación que existe entre la frecuencia, la recencia y el número esperado de compras.

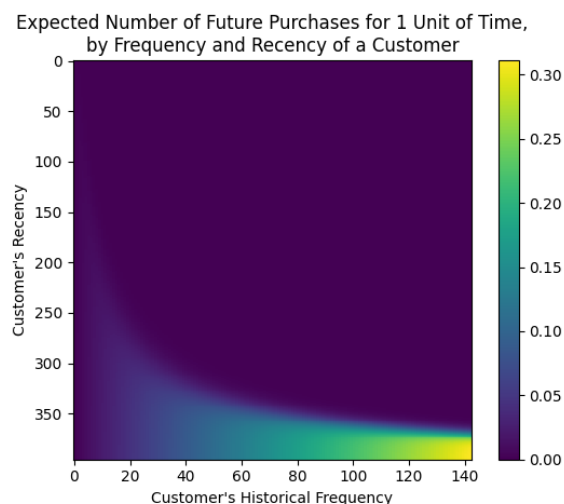


Figura 5.1: Matriz de Frecuencia, Recencia y Compras Esperadas

Se puede apreciar que si un cliente ha interactuado más de 120 veces, y su última interacción la realizó luego de 350 días de estar activo, entonces se puede considerar como uno de los mejores clientes (abajo a la derecha). Los clientes menos activos son los que se encuentran en la esquina superior derecha: compraron mucho rápido y no lo han vuelto a hacer en semanas.

También está la “cola” alrededor de las coordenadas (20, 300). Eso representa al cliente que compra con poca frecuencia, pero que se ha visto recientemente, por lo que podría volver a comprar; no se tiene seguridad de si ha dejado de ser cliente o solo está en un periodo entre compras.

Matriz de Frecuencia, Recencia y Probabilidad de vida

Otra información interesante a evaluar es la relación entre la frecuencia, recencia y la probabilidad de vida de un cliente.

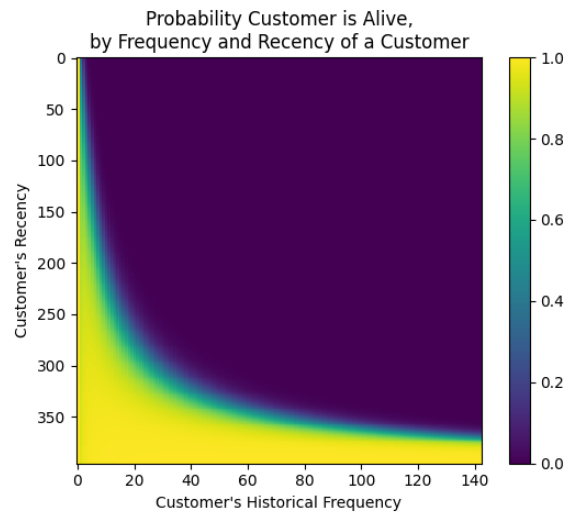


Figura 5.2: Matriz de Frecuencia, Recencia y Probabilidad de vida

De forma similar a la matriz anterior, si los clientes tienen una frecuencia alta de compra y se han visto durante tiempos prolongados, su probabilidad de ser clientes es muy alta (abajo y a la derecha). De igual manera si un cliente se ha visto recientemente tiene una alta probabilidad de estar activo (arriba a la izquierda).

Evaluación del ajuste del modelo

En el siguiente gráfico se evalúan los parámetros de ajuste del modelo, para ello se comparan los datos que se utilizaron para ajustar el modelo con datos artificiales simulados con los parámetros del modelo ajustado.

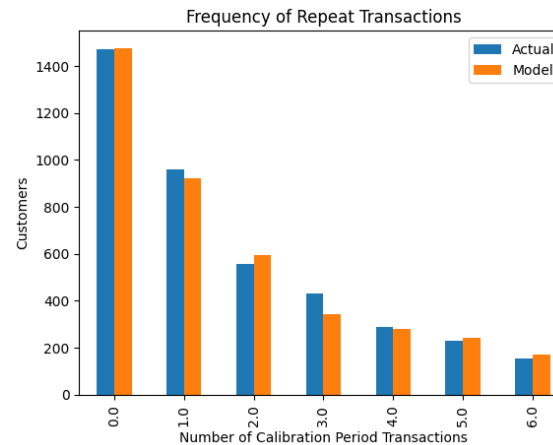


Figura 5.3: Comparación entre el modelo y datos generados

Se puede observar que los datos reales y los datos simulados se alinean bien, esto prueba que el modelo es bueno y representa muy bien el comportamiento de los clientes.

Validación Cruzada

Se puede dividir el conjunto de datos en un grupo de datos de período de calibración y otro grupo de datos de reserva. Esto es importante ya que se quiere probar cómo funciona el modelo en datos que aún no se han visto. La biblioteca `lifetimes` contiene una función para particionar el conjunto de datos:

```
1 summary_cal_holdout = calibration_and_holdout_data(  
    transactional_data, 'CustomerNo', 'Date',  
        calibration_period_end='2019-06-09',  
    observation_period_end='2019-12-09')
```

Listing 5.1: Validación en modelo.ipynb

Luego de dividir la data, se procede a ajustar el modelo con los datos del periodo de calibración:

```
1 model.fit(summary_cal_holdout['frequency_cal'], summary_cal_holdout [  
    'recency_cal'], summary_cal_holdout['T_cal'])
```

Listing 5.2: Validación en modelo.ipynb

Finalmente se compara las compras del periodo de reserva, con las compras esperadas por el modelo en ese mismo periodo:

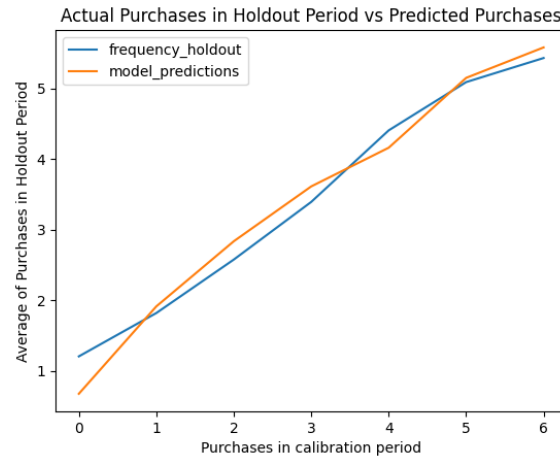


Figura 5.4: Comparación entre compras predecidas y compras reales

Se puede apreciar una gran similitud entre las compras reales del periodo donde el modelo no tiene conocimiento y las compras que el modelo predijo, viendo esto y las pruebas anteriores podemos concluir que el modelo tiene un rendimiento suficientemente bueno para ser usado en el sistema.

5.1.2. API

Para verificar que la API cumple los requerimientos, se realizaron pruebas mediante un software para hacer peticiones HTTP llamado Insomnia, con el se realizaron varias llamadas al recurso `/conditional-probability-alive` con diferentes parámetros de entrada, esto con la finalidad de verificar que la respuesta cumple con los objetivos.

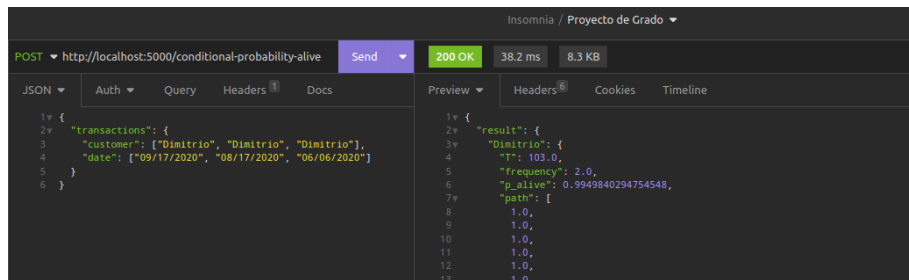


Figura 5.5: Prueba con datos válidos

En el primer caso se envió una petición válida con información sobre varias transacciones y el recurso responde correctamente a la solicitud.

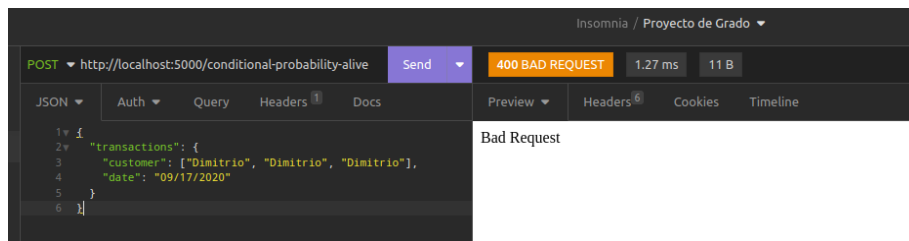


Figura 5.6: Prueba con datos inválidos

En el segundo caso se envió una petición inválida a la API y esta responde con el mensaje adecuado.

5.1.3. Aplicación Web

Ya que el listado de transacciones debe tener un formato específico, se validó la entrada del usuario para garantizar esta estructura, para ello se muestra un texto de ayuda y mensajes con información sobre los diferentes errores de entrada posibles.



DESARROLLO DE UN SISTEMA PARA INFERIR LA DESERCIÓN DE LOS CLIENTES EN UN E-COMMERCE

Este sistema utiliza el modelo BG/NBD para inferir la deserción o churn de un cliente según sus transacciones. El modelo se ajustó utilizando [de transacciones de ventas de una tienda de comercio electrónico con sede en el Reino Unido](#).

Para inferir la deserción de uno o varios clientes, es necesario saber las transacciones que realizaron, por favor registre las transacciones mediante el siguiente formulario:

Agregar Transacciones

Fecha de la Transacción

12/12/2020

Por favor ingrese una fecha válida en el formato DD-MM-YYYY, por ejemplo: 21-02-2020

Identificador de Cliente

Type here

Por favor ingrese un identificador de cliente

REGISTRAR

Figura 5.7: Interfaz con errores de validación

5.2. Resultados

En esta sección se muestra el funcionamiento del modelo obtenido, las predicciones que arrojó según ciertos parámetros de ejemplo y la interpretación de estos resultados.

5.2.1. Modelo

A continuación se van a presentar ejemplos de clientes dentro del dataset, sus resultados y un análisis de la información obtenida:

Ejemplo 1

En este ejemplo se aprecia el comportamiento de un cliente que realizó dos compras consecutivas, al inicio la probabilidad de ser considerado un cliente baja con una mayor rapidez, luego de que el cliente realiza otra compra se puede ver que la

probabilidad ahora disminuye más lentamente. Finalmente luego de la cuarta compra se puede observar que este cliente tiene una relación saludable con la tienda y tiene un patrón de compra definido.

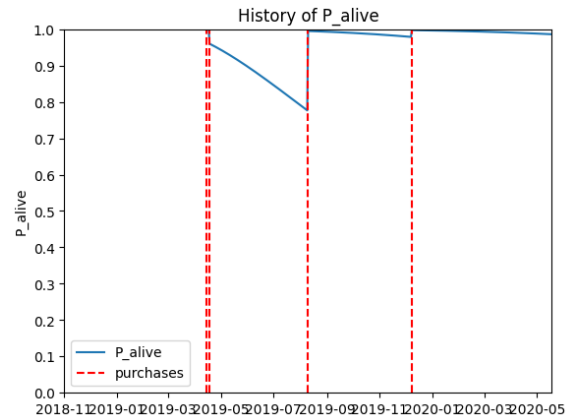


Figura 5.8: Probabilidad de vida Ejemplo 1

Ejemplo 3

Este caso es de interés ya que se puede apreciar a un cliente que compra muy seguido durante todo el año, luego de pasar unos días sin comprar, el modelo penaliza fuertemente su probabilidad de seguir siendo cliente ya que no mantiene su comportamiento anterior.



Figura 5.9: Probabilidad de vida Ejemplo 3

Ejemplo 4

En este ejemplo el cliente realizó dos compras y durante el año la probabilidad de vida del cliente fue disminuyendo pero no fue descartado, como vemos luego el cliente regresó y se completó su patrón de compra.

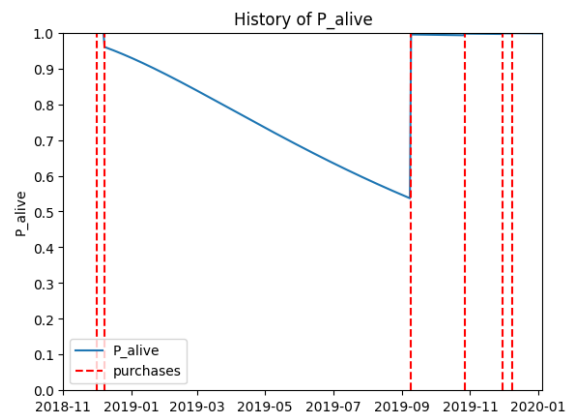


Figura 5.10: Probabilidad de vida Ejemplo 4

5.2.2. API y Aplicación Web

A continuación se puede observar los resultados obtenidos al utilizar el sistema para predecir la probabilidad de deserción de un cliente dada una lista de transacciones del mismo.

Esta prueba verifica que la integración entre la aplicación web y la API funciona correctamente y que el sistema está cumpliendo con la funcionalidad esperada.

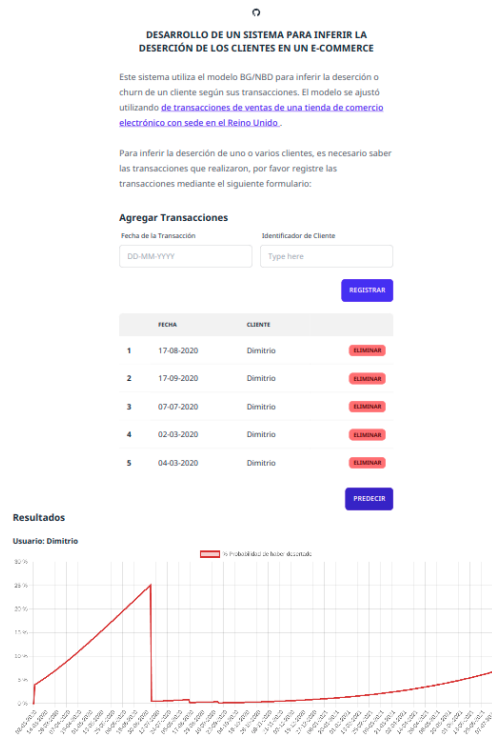


Figura 5.11: Interfaz con resultados

Capítulo 6

Conclusiones y Recomendaciones

En este proyecto se logró definir, diseñar e implementar un sistema con el fin de predecir la deserción de clientes de una tienda de comercio electrónico, para esto se ejecutó una metodología de desarrollo ágil que consistía en un conjunto de requerimientos, los cuales tenían como finalidad cumplir los objetivos propuestos para el proyecto y que luego fueron desglosados en un backlog de tareas, para finalmente ser implementadas mediante un tablero Kanban.

Todo comenzó con una investigación para definir el modelo correcto que posibilitara predecir la deserción de clientes dado un listado de transacciones, concluyendo así en un modelo de la familia “Buy Till You Die” llamado BG / NBD, luego se examinó la web para obtener un dataset que permitiese ajustar los parámetros del modelo seleccionado.

Al tener el modelo se definió una arquitectura del sistema, se implementó cada uno de los módulos y se desplegó en varios servidores para que pueda ser usado. En base a los resultados obtenidos se puede observar que, efectivamente el sistema logra predecir la deserción de clientes en un ambiente continuo y de carácter no contractual como lo es una tienda de comercio electrónico.

El sistema desarrollado en este proyecto puede servir fácilmente como base para futuros proyectos enfocados en mejorar la experiencia de los clientes en ambientes no contractuales, y así identificar oportunidades de re-conversión, de ventas de valor agregado y optimizar los objetivos de negocios de las empresas que trabajen en el ámbito del e-commerce.

Como recomendaciones encontradas durante el desarrollo de este sistema, hemos planteado las siguientes:

- Para obtener mejores resultados, sería ideal que la data utilizada para la construcción del modelo sea el historial de transacciones real de la tienda de comercio electrónico que lo usaría, esto debido a que el comportamiento de los clientes suele variar según la categoría de productos que se venden, la región o país del mundo, el tipo de clientes y otros factores que puedan influir en las decisiones de compra.
- Tomando en cuenta lo anterior, se recomienda integrar el sistema como una extensión o herramienta en plataformas de comercio electrónico populares como Magento, WooCommerce, Vendure, Shopify, etc.
- Si se usaran los datos de compra y clientes de una tienda de comercio electrónico activa, sería necesario actualizar el modelo de vez en cuando, como sugerencia se puede construir el modelo anualmente.
- Para el despliegue del sistema se recomienda utilizar un servicio de despliegue de código automatizado como Heroku, Digital Ocean Apps, Railway o configurar un servidor que haga uso del sistema operativo Linux.

Bibliografía

Neda Abdolvand, Amir Albadvi, y Hamidreza Koosha. Customer lifetime value: Literature scoping map, and an agenda for future research. 2021. URL <https://ssrn.com/abstract=3926592>.

Peter Fader, Bruce Hardie, y Ka Lee. “counting your customers” the easy way: An alternative to the pareto/nbd model. *Marketing Science*, 24:275–284, 2005. doi:10.1287/mksc.1040.0098.

Jesús Falla. Predicción de abandono de clientes en telecomunicaciones mediante el aprendizaje automático. 2021. URL <http://hdl.handle.net/20.500.12010/22247>.

Wei Fu. Research on the construction of early warning model of customer churn on e-commerce platform. *Applied Mathematics and Nonlinear Sciences*, 0(0), 2022. doi:doi:10.2478/amns.2022.1.00016. URL <https://doi.org/10.2478/amns.2022.1.00016>.

Petra Jílková y Petra Králová. Digital consumer behaviour and ecommerce trends during the covid-19 crisis. 2021. URL <https://doi.org/10.1007/s11294-021-09817-4>.

Dorel Mihai Paraschiv, Emilia Țițan, Daniela Ioana Manea, Crina Dana Ionescu, Mihaela Mihai, y Octavian Șerban. The change in e-commerce in the context of the coronavirus pandemic. *Management and Marketing*, 17(2):220–233, 2022. doi:doi:10.2478/mmcks-2022-0012. URL <https://doi.org/10.2478/mmcks-2022-0012>.

David C. Schmittlein, Donald G. Morrison, y Richard Colombo. Counting your custo-

- mers: Who are they and what will they do next? *Management Science*, 33(1):1–24, 1987. ISSN 00251909, 15265501. URL <http://www.jstor.org/stable/2631608>.
- Daeho Seo, Soobin Choi, y Yongmin Yoo. Prevention of customer churn due to issuance of real-time coupons based on deep learning. 2022.
- Ashish Sinha y Sumesh Raizada. Modelling customer churn rate and its use for customer retention planning. 2022. URL <http://dx.doi.org/10.2139/ssrn.3998408>.