



GRADO EN ESTADÍSTICA

— TRABAJO FIN DE GRADO —

*Modelización estadística
de ventas
en el sector retail*

Marta Venegas Pardo

Tutor:

Jose Luis Pino Mejías
Departamento de Estadística e Investigación Operativa

Sevilla, Junio de 2022

Índice general

Prólogo	V
Resumen	VII
Abstract	VIII
Introducción	IX
Índice de Figuras	XI
Índice de Tablas	XIII
1. La ciencia de datos en el sector retail	1
1.1. Modelización estadística de ventas en el sector retail	1
1.2. La Minería de Datos y el Sector Retail	2
2. Análisis de cesta de la compra (Market basket análisis)	5
2.1. Técnica	6
2.2. Medidas de asociación (Indicadores de las reglas de asociación)	6
2.3. Algoritmo a priori	7
3. Modelos estadísticos clásicos	9
3.1. Modelo de Regresión Lineal Generalizado	9
3.1.1. Componentes del Modelo Lineal Generalizado	10
3.1.1.1. Componente aleatoria	10
3.1.1.2. Componente sistemática	11
3.1.1.3. Función link o función enlace	11
3.1.1.4. Estimación de parámetros	11
3.1.2. Modelo de Regresión Poisson	12
3.1.3. Modelo de Regresión Binomial Negativa	13
3.2. Análisis de series temporales	14
3.2.1. Metodología Box-Jenks	16
4. Proceso de la ciencia de datos (Data science process)	19
4.1. Introducción	19
4.2. Etapas del proceso de la ciencia de datos	19
4.2.1. Conocimiento del negocio (Knowledge of Business)	20
4.2.2. Adquisición de los datos (Collect the data)	20
4.2.3. Preparación de los datos (data preparation)	20
4.2.4. Análisis exploratorio de datos (EDA)	20
4.2.5. Modelado	21
4.2.5.1. Máquinas de vector soporte (Support Vector Machines SVMs)	22

4.2.5.1.1.	Descripción del algoritmo	22
4.2.5.2.	K-Nearest Neighbor Regression (KNN)	24
4.2.5.2.1.	Descripción del algoritmo	24
4.2.5.2.2.	Elección del parámetro k	25
4.2.5.3.	Árboles de decisión (XGBoost Model)	26
4.2.5.3.1.	Descripción del algoritmo	28
4.2.5.4.	Evaluación y presentación de resultados (+análisis del error)	28
5. Caso práctico con datos reales		29
5.1.	Análisis de cesta de la compra para productos lácteos	29
5.1.1.	Introducción	29
5.1.2.	Lectura y descripción de los datos	30
5.1.3.	Análisis de ventas	31
5.1.4.	Aplicación del algoritmo a priori	34
5.1.4.1.	Itemsets	34
5.1.4.2.	Reglas de asociación	36
5.1.4.3.	Evaluación de las reglas	39
5.1.4.4.	Reglas maximales	41
5.1.4.5.	Reglas redundantes	42
5.1.5.	Conclusiones	43
5.2.	Proceso de la ciencia de datos	43
5.2.1.	Lectura y descripción de los datos	43
5.2.2.	Preparación de los datos (Preprocesado)	44
5.2.2.1.	Transformación de los datos	44
5.2.2.2.	Duplicados	44
5.2.2.3.	Datos faltantes	44
5.2.2.4.	Outliers	45
5.2.2.5.	Creación de variables	45
5.2.2.6.	Datasets de entrada de modelos	46
5.2.3.	Ánalysis exploratorio de datos (EDA)	47
5.2.3.1.	Resumen de los datos	47
5.2.3.2.	Representaciones gráficas	48
5.2.3.2.1.	Variable precio	51
5.2.3.3.	Grado de asociación de las variables	52
5.2.4.	Modelado	53
5.2.4.1.	Modelos estadísticos clásicos	53
5.2.4.1.1.	Modelo de Regresión de Poisson	53
5.2.4.1.2.	Modelo de Regresión Binomial Negativa	59
5.2.4.1.3.	Conclusiones	66
5.2.4.1.4.	Análisis de Series Temporales	66
5.2.4.2.	Técnicas de aprendizaje automático	76
5.2.4.2.1.	División de los datos en entrenamiento y testeo	77
5.2.4.2.2.	Predicción del volumen total de ventas	77
5.2.4.2.3.	Predicción del volumen de ventas del producto con calcio	83
5.2.4.2.4.	Predicción del volumen de ventas del producto sin calcio	88
5.2.4.2.5.	Comparación de resultados del modelado	94

Conclusiones	96
A. Apéndice: App shiny	97
Bibliografía	102

Prólogo

Este proyecto despertó gran interés en mi desde su inicio, ya que el campo del análisis de datos es aquel hacia el que quiero orientar mi trayectoria profesional.

Me gustaría dedicar estas páginas especialmente a mi madre, María Jesús, a mi abuela Eme, a mi hermana Carlota y a todos los que han estado conmigo estos años, por su confianza en mí, su paciencia y por estar ahí siempre. También quería agradecer a mi tutor su ayuda y orientación en el desarrollo de este trabajo. No quería olvidarme de otros dos grandes profesores, Teresa y Antonio, que desde los primeros cursos hicieron que me apasionara lo que he estudiado.

Resumen

En el presente trabajo fin de grado, titulado *Modelización de ventas en el sector retail* se hace una revisión de diferentes modelos estadísticos y algoritmos de aprendizaje automático con el principal objetivo de aplicarlos a un conjunto de datos real para predecir el volumen de ventas de una pareja de productos.

Para llegar a la fase del modelado, es necesario de realizar un proceso de ciencia de datos completo, que comprende desde el pre-procesado y depuración de los datos, hasta un análisis exploratorio para entender como varían las ventas en función del tiempo.

También se ha incluído una revisión teórica con la correspondiente aplicación práctica de la técnica del análisis de cesta de la compra, aplicado como sistema de recomendación por numerosas empresas a través de internet.

En definitiva, lo que se busca es comprender y modelar el comportamiento de compra del consumidor, para poder así tomar decisiones de negocio que puedan contribuir, por ejemplo, al aumento de la facturación anual y del margen de beneficios.

Abstract

The current essay presents a review of some statistical modeling tools as well as machine learning algorithms in order to develop predictive models with the purpose of predicting daily sales. The article examines a particular market in the retail industry, the market for dairy products in supermarkets.

In order to develop these models, it is necessary to complete an entire data science process from pre-processing and data cleaning to an exploratory data analysis in order to understand how sales vary over time.

This study also develops a theoretical review with the corresponding practical application of a process called *market basket analysis*, the search for meaningful associations in customer purchase behavior. This technique is applied as a recommendation system by numerous companies through the internet.

In essence, what is sought is to understand and model consumer's purchasing behavior, in order to be able to make business decisions that can contribute, to an increase in annual turnover and profit margins.

Introducción

El concepto de retail es una orientación de la dirección del negocio que sostiene que las tareas clave de un minorista son:

- Determinar las necesidades y deseos de su mercado objetivo
- Dirigir la empresa hacia la satisfacción de esas necesidades y deseos de forma más eficiente que sus competidores (Vigaray, 2005).

El comercio detallista o minorista es el último eslabón de la distribución comercial, siendo el intermediario que se dedica a la venta de productos, bienes o servicios a los consumidores o usuarios finales (Burrueto, 1999).

Este sector aglutina a comerciantes y empresas encargadas de la comercialización, ofreciendo de una gran variedad de productos y servicios a los consumidores. Una tienda, un supermercado o una librería son claros ejemplos de lo que es el sector retail.

A continuación podemos destacar los siguientes objetivos de este estudio:

- Analizar y entender como varía la demanda de productos en función del tiempo y de otros factores
- Descubrir asociaciones y patrones entre productos aplicando un análisis de cesta de la compra
- Investigar y revisar diversos algoritmos de aprendizaje automático y aprendizaje estadístico para modelar las ventas de los diferentes productos
- Utilizar Rstudio, un entorno de desarrollo integrado para el lenguaje de programación R como soporte al estudio estadístico de los datos

Para ello, se llevará a cabo una revisión teórica de técnicas para la modelización de la variable *volumen de ventas* para posteriormente aplicarlas a un caso práctico real de ventas de productos lácteos, con el propósito final de construir un modelo que ayude a predecir la demanda futura.

Dentro de las técnicas que se van a exponer a lo largo del trabajo, podemos distinguir dos vertientes: las técnicas puramente de aprendizaje estadístico y técnicas de aprendizaje automático (Machine Learning).

El término *Machine Learning* (ML, Aprendizaje Automático) se utiliza en el campo de la Inteligencia artificiál para referirse a algoritmos de predicción. Muchas de estas técnicas provienen del campo de la Estadística y por tanto, esta rama aplicada de las Matemáticas es la base de todos estos modelos para analizar datos. Por otro lado, desde el campo de la Estadística Computacional, se introdujo el término *Statistical Learning* (AE, Aprendizaje Estadístico) para referirse a este tipo de herramientas desde un punto de vista estadístico, es decir, se tiene en cuenta la incertidumbre debida a no disponer de toda la información.

Además, el ML no se preocupa del origen de los datos, siendo frecuente la consideración de un conjunto enorme de datos, lo que equivale a disponer toda la información (la población completa). Por el contrario, en el caso del AE, se trata de comprender la estructura de los datos y si son representativos de la población de interés.

Siguiendo esta línea, en el año 2001, Leo Breiman publica *Modelos Estadísticos*, donde diferencia dos objetivos en el análisis de datos, que el define como *información* y *predicción*. Cada uno de ellos da lugar a una cultura en el uso de modelos estadísticos para llegar a conclusiones a partir de los datos:

- *Modelización de datos*: se trata del desarrollo de modelos estocásticos que permitan ajustar los datos y realizar inferencia.
- *Modelización algorítmica* (en sentido predictivo): esta cultura está interesada en los algoritmos de predicción, no en los mecanismos que generan los datos, siendo el ML la base de esta cultura.

Para tratar de cumplir los objetivos, se ha estructurado el trabajo en diferentes capítulos. Los primeros cuatro capítulos consisten en la revisión teórica de todas las técnicas que posteriormente aplicaremos a un caso práctico. En primer lugar, un capítulo introductorio sobre el proceso de la ciencia de datos en el sector retail, a continuación, un segundo capítulo para la explicación de lo que se conoce como análisis de cesta de la compra, que tiene como objetivo conocer las asociaciones entre diversos productos. Los capítulos tercero y cuarto explican respectivamente, las técnicas clásicas de modelado estadístico y el proceso de ciencia de datos completo con las correspondientes técnicas de aprendizaje automático. Un quinto capítulo donde se aplicarán todas las técnicas estudiadas a un caso práctico con datos reales. Por último, en el capítulo sexto se exponen las conclusiones extraídas y se analizan los objetivos iniciales.

Índice de figuras

4.1.	División de datos muestrales para entrenamiento, validación y testeo	21
4.2.	Vectores de Soporte de Regresión.	24
4.3.	Esquema conceptual del algoritmo KNN de regresión.	25
4.4.	Árbol de decisión. Partes	26
4.5.	Algoritmo XGBoost	28
5.1.	Matríz de correlación. Fuente: elaboración propia con datos de ventas.	52
A.1.	Barra de navegación. Fuente: elaboración propia con Shiny.	97
A.2.	Carrusel interactivo. Página de inicio. Fuente: elaboración propia con Shiny.	98
A.3.	Índice de contenidos. Página de inicio. Fuente: elaboración propia con Shiny.	98
A.4.	Página de análisis de cesta de la compra Fuente: elaboración propia con Shiny.	99

Índice de tablas

5.1.	Productos más frecuentes	32
5.2.	Importe medio de venta y precio por producto.	50
5.3.	Métricas del mejor modelo SVM. Algoritmo 1.1	77
5.4.	Métricas en el remuestreo. Modelo SVM. Algoritmo 1.1	78
5.5.	Métricas del mejor modelo KNN. Algoritmo 2.1	79
5.6.	Métricas en el remuestreo. Modelo KNN. Algoritmo 2.1	79
5.7.	Métricas del mejor modelo XGBoost. Algoritmo 3.1	80
5.8.	Métricas en el remuestreo. Modelo XGBoost. Algoritmo 3.1	80
5.9.	Resultados de predicción. Volumen total de ventas. Algoritmo Máquina de vector soporte	82
5.10.	Métricas del mejor modelo SVM. Algoritmo 1.2	83
5.11.	Métricas en el remuestreo. Modelo SVM. Algoritmo 1.2	83
5.12.	Métricas del mejor modelo KNN. Algoritmo 2.2	85
5.13.	Métricas en el remuestreo. Modelo KNN. Algoritmo 2.2	85
5.14.	Métricas del mejor modelo XGBoost. Algoritmo 3.2	86
5.15.	Métricas en el remuestreo. Modelo XGBoost. Algoritmo 3.2	86
5.16.	Resultados de predicción. Volumen total de ventas. Algoritmo XGBoost .	87
5.17.	Métricas del mejor modelo SVM. Algoritmo 3.1	89
5.18.	Métricas en el remuestreo. Modelo SVM. Algoritmo 3.1	89
5.19.	Métricas del mejor modelo KNN. Algoritmo 3.2	91
5.20.	Métricas en el remuestreo. Modelo KNN. Algoritmo 3.2	91
5.21.	Métricas del mejor modelo XGBoost. Algoritmo 3.3	92
5.22.	Métricas en el remuestreo. Modelo XGBoost. Algoritmo 3.3	92
5.23.	Resultados de predicción. Volumen total de ventas. Algoritmo máquina de vector soporte	93
5.24.	Resultados de la predicción en los datos de testeо. Comparación de métricas.	94

Capítulo 1

La ciencia de datos en el sector retail

1.1. Modelización estadística de ventas en el sector retail

En la actualidad gran cantidad de empresas y organizaciones almacenan una cantidad creciente de datos de cualquier actividad que desempeñen. Sin embargo, no todos los datos son luego procesados, es por ello que se busca implementar herramientas que ayuden a procesar estos datos y poder así transformarlos en información útil para el negocio que ayude en la toma de decisiones, ya que en muchas empresas, hasta hace unos años, estas decisiones eran tomadas de manera intuitiva o por conocimientos históricos que poseía la empresa debido a situaciones previas.

Se pueden aplicar diferentes técnicas para que empresas del sector retail se vean beneficiadas al conocer los patrones de compra de sus clientes generando así indicadores para poder llevar un control de la información y que esta sea de utilidad para la mejor toma de decisiones haciendo uso de diferentes herramientas como por ejemplo la Minería de Datos (*Data Mining*), que es una tecnología computarizada encargada de extraer, recoger, depurar y modelar grandes volúmenes de datos para así encontrar tendencias y patrones de coincidencia entre los datos que inicialmente eran desconocidos. Por tanto, la minería de datos es capaz de extraer información útil de los datos que poseen las empresas, y es por tal motivo que el usar esta herramienta con éxito proporcionará a la empresa una ventaja frente a sus competidores.

Por ejemplo, una tienda que ofrezca a sus clientes un servicio en línea, podrá tener acceso a todos los datos de los consumidores de sus servicios y/o productos. Esta información les servirá para conocer a sus clientes, sus preferencias y tendencias, pudiendo ofrecerle un servicio más adecuado o personalizado.

Entre las diferentes técnicas de la minería de datos, podemos encontrar: redes neuronales, regresión lineal, árboles de decisión, reglas de asociación, agrupamiento, análisis factorial, series temporales y pronóstico (forecasting). De todas ellas, las *reglas de asociación* son una herramienta muy potente, ya que encuentra los diferentes hechos que tienen en común un conjunto de datos y los asocia. Además, también es útil para encontrar asociaciones y/o correlaciones entre los datos. Esta técnica puede ser muy útil a la hora de poner precios a los productos y para desarrollar distintas herramientas de marketing para llegar al cliente con éxito.

La minería de datos es una herramienta aplicada por muchas empresas del sector retail, perteneciendo a muchos sectores como las finanzas, salud, bancos, servicios públicos y seguros; ya que todos estos sectores tienen una gran cantidad de datos referidos a sus clientes, proveedores, productos y/o servicios.

Estos datos pueden ser de tipo demográfico (como edad, sexo o estado civil), datos económicos (carrera, puesto de trabajo, ingresos por familia), datos geográficos (país, ciudad de residencia, dirección,...) o incluso proceder de redes sociales o de las compañías de telecomunicaciones y exigir el uso de técnicas de Big Data.

A continuación, vamos a exponer dos ejemplos de empresas del sector Retail que aplican la Minería de Datos:

■ **Wal Mart**

Se trata de corporación multinacional de tiendas de origen estadounidense, que vende al por menor y funciona como una cadena de hipermercados, almacenes grandes de descuento y almacenes de comestibles.

Esta multinacional es una de las pioneras en el uso de la minería de datos y la gestión de sus datos. Toma datos sobre transacciones que se realizan en sus 2900 tiendas, almacenándolos en una base de datos con una capacidad de 1.5 terabytes. La empresa ofrece a sus proveedores información sobre los productos, para que éstos puedan identificar los patrones de compra de sus clientes, logrando así la gestión de los inventarios en el almacén.

■ **Papas “Chips”**

Es una empresa que distribuye productos como golosinas y refrescos. La empresa utiliza la minería de datos para ofrecer a sus clientes un buen servicio, obteniendo así la fidelización de los mismos, lo cual repercute positivamente en los ingresos de la empresa.

La empresa aplica técnicas de minería de datos para llevar un registro de ventas, conociendo así los períodos con un volumen de ventas mayor para un producto o productos determinados. De esta forma, se logra abastecer a los clientes teniendo siempre productos en stock.

1.2. La Minería de Datos y el Sector Retail

Como ya sabemos, Retail hace referencia al detalle de productos, por tanto, se trata de un sector empresarial enfocado a productos que se venden de manera masiva, por lo que es un sector con una gran cantidad de clientes, dado que es el contacto directo con el consumidor final de cada producto ofrecido en el mercado.

La Minería de Datos y el Sector Retail están relacionados dependiendo del tipo de la empresa en cuestión, su tamaño o el tipo de cliente (consumidores, familias, minoristas, supermercados, centros comerciales, bancos, pequeños establecimientos de venta como tiendas,...), por tanto, esta herramienta es útil para analizar todo tipo de bases de datos con el objetivo de obtener información sobre los clientes, segmentarlos en función de sus necesidades, tendencias de compra, ...

Al aplicar la minería de datos en este sector, se pueden obtener resultados como los patrones de compra, las preferencias de los clientes o asociaciones entre productos que no

conocíamos, abriendo la posibilidad a la creación de distintas estrategias como ofertas o packs.

Por tanto, como resultado de la aplicación de la minería de datos en el sector retail, obtendremos información relevante para cada empresa, ya que nos mostrará el comportamiento de cada tipo de cliente que tenga cada empresa mediante la aplicación de diferentes técnicas y herramientas estadísticas.

Capítulo 2

Análisis de cesta de la compra (Market basket análisis)

En este capítulo revisaremos el método de análisis de cesta de la compra con el fin de identificar patrones o asociaciones entre diversos grupos de productos. Esta metodología pretende, además de la validación de distintas asociaciones que se pueden considerar obvias debido a una reiterada compra conjunta, encontrar relaciones entre productos cuya asociación no es tan evidente. La principal fuente de información para llevar a cabo este tipo de análisis serán datos que recogen las transacciones de los clientes en cada compra.

El concepto de reglas de asociación fué introducido en el año 1993 por la publicación del artículo de Agrawal, R; Imielinski, T.; Swami, A. *Mining association rules between sets of items in large databases*.

A continuación vamos a definir una serie de conceptos para entender mejor este método. Una *asociación* es una concurrencia de dos o más cosas. Por ejemplo, los perritos calientes pueden estar asociados con los refrescos, los cebolla o el ketchup. Se dice que existe una asociación *positiva* si la presencia de algunos productos implica la presencia de otros dentro de la misma transacción. También existe la asociación *negativa*, la cual consiste en que la presencia de algunos productos haga muy improbable la presencia de otros elementos en la misma transacción.

Se conoce como *reglas de asociación* a la agrupación de productos en función de la afinidad existente entre ellos. La utilidad de estas reglas se encuentra en la identificación de oportunidades y el diseño de diversos grupos de productos que puedan ser atractivos para los consumidores. Estas reglas tienen como finalidad el descubrimiento de las relaciones implícitas en los datos. Cada una de estas reglas consta de un antecedente y un consecuente. Por ejemplo, en la regla siguiente regla de asociación: Si un consumidor compra un Perrito caliente, también tiende a comprar un refresco, cebolla y ketchup. Aquí el Perrito caliente es el antecedente, y el refresco, la cebolla y el ketchup son los consecuentes.

Cuando aplicamos un análisis de cesta de la compra a unos datos, habitualmente podemos obtener tres posibles resultados generados a partir de las reglas de asociación:

- **Resultados factibles:** Contienen información útil y de calidad, ya que los patrones identificados son factibles. Estas reglas nos podrían ayudar en la toma decisiones como por ejemplo, saber a qué productos se aplicar una promoción con el objetivo de impulsar las ventas de los mismos y de sus productos relacionados.

- **Resultados triviales:** Estos son conocidos por aquellos que están familiarizados con el negocio en cuestión. Este tipo de reglas nos muestra productos en los que no se plantea la compra de un producto sin otro, por ejemplo, la compra de pintura requiere la necesidad de comprar también pinceles.
- **Resultados imposibles:** Estos resultados implican la obtención de reglas incongruentes, ya que nos proporciona un información que no facilita el entender el comportamiento del consumidor, sino que se trata de casos puntuales en un momento determinado.

2.1. Técnica

Podemos expresar las reglas de asociación de la siguiente forma: $A \rightarrow B$, donde A e B son conjuntos de ítems. Sea $I = \{i_1, i_2, \dots, i_m\}$ ítems y sea D una serie de transacciones, siendo T una transacción. Se tiene que $T \subseteq I$, y por tanto, el conjunto I tendrá que ser redefinido como binario $\{0, 1\}$. La secuencia de valores de cada transacción T , será generada a partir de la identificación de atributos de I con valor 1.

Se conoce como itemset A , al conjunto de artículos $A \subset I$, por tanto, cada transacción T contiene contiene al itemset A , $A \subseteq T$. Por tanto, una regla de asociación es una implicación de la forma:

$$A \rightarrow B, A \subset I, B \subset I, A \cap B = \emptyset$$

2.2. Medidas de asociación (Indicadores de las reglas de asociación)

Una vez obtenidas las reglas de asociación, es necesario evaluarlas. Dada la siguiente regla de asociación $A \rightarrow B$, podemos definir tres indicadores: lift, soporte y confianza. La importancia de la obtención de éstos radica en la información no redundante que aportan sobre el conjunto de reglas obtenidas

1. **Lift:** Este parámetro indica si realmente existe una asociación entre uno o varios productos y de qué tipo es, si positiva o negativa. Su valor viene dado por:

$$\text{lift} = \frac{P(B|A)}{P(B)P(A)}$$

Es decir, la probabilidad de comprar dos ítems a la vez (asociación) entre la probabilidad de comprarlos por separado. Por eso, podemos decir que este parámetro indica la fuerza de asociación entre los productos de la izquierda y de la derecha de la regla, y cuanto mayor sea su valor, mayor será el vínculo entre los productos.

Este parámetro puede ofrecernos tres posibles resultados:

- Si $\text{lift} > 1$, la probabilidad de que ambos productos estén asociados es mayor que la probabilidad de que no lo estén, considerando que éstos guardan una relación positiva.

- Si $lift \approx 1$, este resultado sugiere que los productos son independientes y su presencia conjunta se debe al azar.
 - Si $lift < 1$, la probabilidad de que los productos no estén asociados es mayor que la probabilidad de que estén, indicando que los productos no están asociados y guardan una relación negativa.
2. **Soporte:** El soporte del elemento A del conjunto de transacciones D viene dado por:

$$supp(A) = P(A) = \frac{\text{Nº transacciones con } A}{\text{Nº total de transacciones}}$$

Es decir, se trata de la proporción de transacciones que contienen una combinación particular de elementos (itemset) en relación con el número total de transacciones. Es un valor que está acotado: $0 \leq supp \leq 1$, y cuánto mayor sea su valor, mayor probabilidad de que la asociación considerada se repita en futuras transacciones.

3. **Confianza:** este indicador nos ofrece información sobre la precisión de la asociación, se trata de la probabilidad de comprar el elemento B , dado que el elemento A aparece en la transacción y se puede calcular de la siguiente forma:

$$C(A \rightarrow B) = P(B|A) = \frac{P(B \cap A)}{P(A)} = \frac{\text{Nº transacciones con } A \text{ y } B}{\text{Nº de transacciones en las que aparece } A}$$

Al igual que el soporte, su valor está acotado: $0 \leq C \leq 1$.

Nota: A la hora de cálculo de estos índices, para el soporte y la confianza es necesario especificar los valores de corte (umbrales) del soporte y la confianza, para así poder concluir si una asociación es o no significativa.

Por último vamos a describir el algoritmo que se va a emplear de cara al análisis de las reglas de asociación, el algoritmo *A-Priori*.

2.3. Algoritmo a priori

El algoritmo *A-Priori* permite generar las reglas de asociación. Para poder aplicarlo, es necesario que las variables sean categóricas, discretizando las variables numéricas. El proceso está basado en el conocimiento derivado a través de las propiedades que se repiten en un conjunto de ítems. Asimismo, se va a considerar como elemento frecuente el que satisface un soporte mínimo para la aplicación del algoritmo.

Este algoritmo, es de tipo iterativo. Cada iteración i genera una serie de elementos candidatos, C_i obtenidos de los datos de las transacciones, para posteriormente comprobar si son frecuentes. Los candidatos que presenten una mayor frecuencia, serán los que se utilizarán para formar el conjunto de candidatos C_{i+1} para la siguiente iteración. Para poder asegurar la generación de candidatos C_{i+1} , se realizarán uniones de conjuntos frecuentes hallados en la iteración anterior, por lo que si hay $i + 1$ elementos comunes, se producirá una unión entre dos conjuntos de elementos, desechando duplicidades. De esta forma, el proceso se detendrá cuando los candidatos de C_{i+1} no son frecuentes en la siguiente iteración.

Capítulo 3

Modelos estadísticos clásicos

A continuación se exponen modelos estadísticos clásicos de cara a predecir la demanda de los productos. Los modelos que se van a revisar son los siguientes: MLG (modelo de Regresión Lineal Generalizado, Modelo de Regresión de Poisson y Modelo de Regresión de Binomial Negativa) y series temporales La variable objetivo depende de varios factores: el período del año, el precio del producto, el precio de los productos competidores o los gustos de cada consumidor, entre otros. Se trata de una variable cuantitativa discreta, ya que el número de productos que se venden será un valor entero no negativo $y = 0, 1, \dots$

3.1. Modelo de Regresión Lineal Generalizado

El objetivo es encontrar un modelo estadístico que describa la situación real de ventas de productos a través de un Modelo Lineal (MLG), donde una variable de interés (variable objetivo) pueda ser descrita por un conjunto de variables explicativas (variables independientes).

Para ello, debemos estimar los parámetros que caracterizan al modelo, es decir, medir el efecto de cada variable explicativa sobre la variable objetivo, y con este fin, es necesario definir una serie de hipótesis del modelo de regresión lineal general:

- Independencia lineal entre las variables explicativas: Esto significa que cada variable explicativa contiene información adicional sobre la variable objetivo, ya que si hubiera información repetida, habría variables explicativas dependientes linealmente de otras.
- Los MLG suponen que existe una función g , llamada función link, que relaciona la media de la variable respuesta, μ con el resto de variables explicativas de la siguiente forma:

$$E[Y] = \mu = g^{-1}(\eta) = g^{-1}(X^t \beta)$$

Siendo:

- Y la variable objetivo
- $E(Y)$ es el valor esperado de la variable Y
- $\eta = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p = X^t \beta$ es el predictor lineal, se trata de una combinación lineal de parámetros desconocidos

- X_1, \dots, X_p son las variables explicativas
- $\beta = (\beta_0, \beta_1, \dots, \beta_p)$ representan el efecto de cada variable independiente sobre la variable objetivo
- g es la función link, monótona y diferenciable

3.1.1. Componentes del Modelo Lineal Generalizado

En este tipo de modelización estadística podemos diferenciar tres componentes: la componenete aleatoria, la sistemática y la función link o enlace. Será la combinación de estas tres componentes la que defina por completo un Modelo Lineal Generalizado.

3.1.1.1. Componente aleatoria

Esta componente es la que identifica la variable respuesta y su distribución de probabilidad.

Sea Y la variable aleatoria objetivo o variable respuesta objeto de estudio y sean las n variables aleatorias independientes e idénticamente distribuidas Y_1, \dots, Y_n la muestra aleatoria procedente de Y . Siendo Y la componente aleatoria cuya distribución pertenece a la familia exponencial de distribuciones.

En los MLG se supone que la variable respuesta Y se distribuye de tal forma que su función de probabilidad, en el caso de estar modelizando una variable discreta o función de densidad para el caso continuo viene dada por la siguiente expresión general, conocida como forma canónica:

$$f(y; \theta, \phi) = a(y, \phi) \cdot e^{\left(\frac{y\theta - k(\theta)}{\phi} \right)}$$

Donde

- θ es el parámetro canónico
- $k(\theta)$ es la función cumulante
- $\phi > 0$ se trata del parámetro de dispersión
- $a(y, \phi)$ es una constante normalizadora
- El soporte no depende de θ ni de ϕ

Además, la media de y es función del parámetro canónico θ , por tanto, se tiene que:

$$E(Y) = \mu = \frac{\partial}{\partial \theta} k(\theta), \text{Var}(Y) = \sigma^2 = \phi \frac{\partial^2}{\partial \theta^2} k(\theta) = \phi \frac{\partial}{\partial \theta} \left(\frac{\partial}{\partial \theta} k(\theta) \right) = \phi \frac{\partial^2 \mu}{\partial \theta^2} > 0.$$

Es decir, μ es una función estrictamente creciente de θ , por lo que estos dos parámetros mantienen una relación biyectiva.

A

$$V(\mu) = \frac{\partial \mu}{\partial \theta}$$

se le denomina función varianza, por lo que se tiene que:

$$V(Y) = \phi V(\mu)$$

3.1.1.2. Componente sistemática

Se trata de la componente que especifica las variables predictoras utilizadas en la función predictora lineal en forma de efectos fijos de un modelo lineal y recoge la variabilidad de la respuesta Y expresada a través de p variables explicativas X_1, \dots, X_p , que denotamos por X , y de los correspondientes parámetros desconocidos $\beta = (\beta_0, \beta_1, \dots, \beta_p)'$.

Esta componente, también conocida como predictor lineal, viene representada por η y es una combinación lineal de las variables explicativas, que viene dada por:

$$\eta = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p = X^t \beta = X^t \beta$$

3.1.1.3. Función link o función enlace

La *función link*: se trata de una función del valor esperado de la variable respuesta $E[Y]$, como una combinación lineal de las variables predictoras. Sin embargo, en muchos casos reales esta relación no es adecuada, por lo que es necesario la introducción de una función con el objetivo de relacionar el valor esperado con las variables explicativas. Por ello, introducimos la función link o función enlace, $g(\cdot)$ que relaciona μ con el predictor lineal de la siguiente forma:

$$g(\mu) = \beta_0 + \beta_1 x_1 + \dots + \beta_p x_p$$

En problemas reales, pueden existir varias funciones link, por lo que se elegirá aquella que facilite la interpretación del modelo óptimo obtenido. En particular, para cada elemento de la familia exponencial existe una función enlace denominada función canónica, que permite relacionar el parámetro canónico con el predictor lineal.

$$\theta_i = \theta(\mu_i) = \eta_i = X_i^t \beta \quad g(\mu_i) = \theta(\mu_i)$$

3.1.1.4. Estimación de parámetros

Tras la construcción de los modelos, se estiman los parámetros desconocidos del predictor lineal, $\beta = (\beta_0, \beta_1, \dots, \beta_p)'$ por $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_p)'$ y el valor del parámetro de dispersión ϕ por $\hat{\phi}$. Posteriormente, se valora la precisión de las estimaciones con el objetivo de seleccionar un modelo óptimo.

Generalmente, la estimación de los parámetros se lleva a cabo por el método de la *Máxima Verosimilitud* o el método de *Mínimos Cuadrados Ordinarios*. Una vez desarrollados los modelos, se realizará una comparación de los mismos con el objetivo de seleccionar el mejor. En el caso del modelado con fines predictivos, se seleccionará el modelo que explique el mayor porcentaje de variabilidad de la respuesta.

Para ello, emplearemos el **Criterio de información de Akaike (AIC)**, medida relativa de la calidad de un modelo estadístico. Este criterio trata de proporcionar una compensación entre la bondad de ajuste del modelo y la complejidad del mismo, es decir, el criterio penaliza al número de parámetros.

Dado un conjunto de modelos candidatos para los datos, el modelo preferido es aquel que tiene mínimo valor del AIC.

En el caso general, el AIC viene dado por la siguiente expresión:

$$AIC = 2k - 2\ln(\hat{L})$$

Siendo:

- k el número de parámetros del modelo
- \hat{L} es el máximo valor de la función de verosimilitud para el modelo estimado

Otro criterio en el que nos basaremos es en el criterio de bondad de ajuste, destacando el cálculo del **coeficiente de determinación R^2** , que es una medida del grado de fiabilidad o bondad de ajuste del modelo ajustado a un conjunto de datos. Se trata de una medida acotada por definición, siendo sus límites $0 \leq R^2 \leq 1$. Un coeficiente de determinación igual a 1 indica un ajuste lineal perfecto, y por tanto, la variación total de la variable Y es explicada por el modelo de regresión. Por el contrario, el valor 0 indica que el modelo no explica nada de la variación total de la variable Y .

Para la bondad de ajuste, otra medida interesante es el RMSE, raíz del error cuadrático medio. Representa la raíz cuadrada de la distancia promedio entre el valor real y el pronosticado e indica el ajuste absoluto del modelo a los datos, es decir, cómo de cerca están los puntos observados de los valores predichos del modelo. Valores más bajos de RMSE indican un mejor ajuste.

En muchos casos la variable respuesta es de tipo conteo, como lo es la variable que queremos modelizar, demanda de productos. Se denominan variables de recuento o variables de tipo conteo, a aquellas que determinan el número de sucesos que ocurren en una misma unidad de observación en un intervalo espacial o temporal definido. Esta variable Y , puede tomar infinitos números de valores y su probabilidad va en descenso a medida que sea mayor el valor de la variable.

Para este caso, los modelos que tienen especial interés y que podemos formalizar a través de modelización lineal son el modelo de *Poisson* y el modelo de *Binomial negativa*. Estos modelos permiten analizar el comportamiento de variables de conteo frente a los valores del conjunto de variables explicativas.

3.1.2. Modelo de Regresión Poisson

Se trata del modelo más simple y es el modelo de referencia para variables respuesta de tipo conteo. Este modelo asume que la variable respuesta Y sigue una distribución de Poisson, por lo que en el caso de la modelización de ventas, se define como el número de ventas que ocurren en un intervalo de tiempo, cuya ocurrencia es aleatoria.

Esta distribución se caracteriza por que la media y varianza coinciden:

$$E(Y) = Var(Y) = \mu$$

Se tiene que la distribución de probabilidad de Poisson, y en nuestro caso, la probabilidad de observar y ventas es:

$$P(Y = y) = \frac{\mu^y e^{-\mu}}{y!}, \quad y = 0, 1, \dots; \mu > 0$$

Y por tanto, la forma canónica o componente aleatoria para esta distribución es la siguiente:

$$f(y; \mu) = e^{-\mu} \cdot \frac{\mu^y}{y!} = \frac{1}{y!} e^{y \log(\mu) - \mu}, \quad y \in \{0, 1, \dots\}$$

Es decir, el modelo de Posición se obtiene tomando como función enlace el parámetro canónico.

donde:

- $\theta = \log(\mu)$ es el parámetro canónico
- $k(\theta) = \mu = e^\theta$ es la función cumulante
- $\phi = 1$ el parámetro de dispersión
- $a(y, \phi) = 1/y!$ la constante normalizadora

En este caso se tiene que: $g(\mu_i) = X_i^t \beta$ y una elección usual de la función link g el parámetro canónico, $g(x) = \log(x)$, lo que equivale a:

$$\mu_i = \exp(\beta_0) \cdot \exp(x_{i1}\beta_1) \dots \exp(x_{ip}\beta_p) = \exp(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \text{ ó } \log(\mu_i) = \eta_i = X_i \beta$$

así si x_i se incrementa en una unidad, entonces μ_i se multiplica por $\exp(\beta_i)$. Por tanto, si $\beta_i > 0$, μ_i crece cuando x_i aumenta y si $\beta_i < 0$, μ_i decrece cuando x_i aumenta.

Este modelo se ha desarrollado suponiendo que la media y la varianza de los datos coinciden (equidispersión). Sin embargo, suele ocurrir lo que se conoce como sobredispersión, es decir, que la varianza es mayor que la media. Lo habitual es que esta situación se de debido a la existencia de heterogeneidad entre las observaciones. Cuando esto ocurra, recurriremos al modelo binomial negativa.

3.1.3. Modelo de Regresión Binomial Negativa

Este modelo es empleado para variables de tipo conteo cuándo existe sobredispersión, es decir, la media condicional es menor que la varianza condicional (no coinciden). Existen diferentes modelos binomiales negativos en función de la variable que se trate de modelar, pero en este trabajo nos centraremos en el caso de datos de tipo conteo.

La distribución binomial negativa estudia la probabilidad de observar un número determinado de fracasos y (no se producen ventas), antes del r -ésimo éxito (se venden r unidades) en una serie de experimentos Bernoulli independientes, siendo r un número positivo. Se tiene que esta distribución pertenece a la familia exponencial si el parámetro de dispersión ϕ es una constante.

Se dice que la variable aleatoria de conteo (número de ventas) Y_i , con $i = 1, \dots, n$ sigue una distribución Binomial Negativa de parámetros r y p , y se representa como $Y_i \sim BN(r, p)$, si su función de probabilidad viene dada por:

$$P[Y_i = y_i] = \left(\frac{y_i + r - 1}{r - 1} \right) p^r (1 - p)^{y_i}$$

donde

- $0 < p < 1$
- $r > 0$

- $y_i = 0, 1, 2, \dots$

Y en este caso, la forma canónica o componente aleatoria para esta distribución es la siguiente:

$$f(y; \mu) = \exp \left\{ y \cdot \ln(1 - p) + r \ln(p) + \ln \left(\frac{y_i + r - 1}{r - 1} \right) \right\}$$

donde

- $0 < p < 1$
- $r = 0, 1, 2, \dots$
- $y_i = 0, 1, 2, \dots$
- $\theta = \log(1 - p)$ es el parámetro canónico
- $k(\theta) = -rln(p) = -rln(1 - e^\theta)$ es la función cumulante

En este caso la función link es de tipo logarítmico y viene dada por:

$$g(\mu_i) = \theta(\mu_i) = \ln \left(\frac{\alpha \mu_i}{1 + \alpha \mu_i} \right) = X_i^t \beta = \eta_i$$

3.2. Análisis de series temporales

Aplicaremos este modelo de predicción para tratar de identificar los patrones de la demanda anterior a lo largo del tiempo y luego proyectar (predecir) los patrones en el futuro.

Se define una serie temporal como una sucesión de datos ordenados en el tiempo que corresponden a una misma variable. Los datos suelen ser tomados en intervalos regulares de tiempo.

Nuestro objetivo dentro del análisis de series temporales será identificar el proceso estocástico que ha sido capaz de generar la serie de estudio.

Se dice proceso estocástico a una colección o familia de variables aleatorias $\{X_t, \text{ con } t \in T\}$ que siguen la misma ley de distribución y están relacionadas entre sí, pudiendo por este motivo, describir la información de estas variables en términos de medias, variaciones y covarianzas.

A continuación encontramos las cuatro etapas en un análisis descriptivo de series temporales para elegir un modelo que se adecue a nuestros datos:

- **Representación gráfica de la serie.** Para tener así una primera aproximación del comportamiento de la serie y la existencia de posibles tendencias.
- **Modelización:** Se trata de encontrar el modelo que mejor se ajuste a los datos.
- **Validación de los modelos:** Es necesario saber si el modelo ajustado es adecuado o no, por lo que es muy importante el estudio de los residuos.
- **Predicciones:** Una vez construido y validado un modelo, realizaremos estimaciones del futuro con nuevas observaciones.

En un enfoque clásico de series temporales, asumiremos que el comportamiento de la variable con respecto al tiempo se compone de cuatro componentes:

1. **Tendencia:** Se trata del movimiento suave y regular de la serie a largo plazo. La tendencia existe cuando hay un aumento o disminución a largo plazo de los datos. Puede ser lineal (ajuste mediante una recta) o no lineal (aproximación mediante una curva, como por ejemplo logarítmica o exponencial)
2. **Ciclo:** Componente de tipo oscilante caracterizada por movimientos recurrentes en torno a la tendencia de la serie y que se repiten cada año pero sin una frecuencia fija.
3. **Componente estacional:** Se trata de movimientos regulares dentro de la serie con una periodicidad menor a un año, es decir, aquello que ocurre generalmente y con la misma intensidad año tras año en los mismos períodos, por ejemplo, en la misma época del año o día de la semana. Vamos a denotar por L al número de estaciones.
4. **Componente irregular:** Se trata de las variaciones de la serie sin un comportamiento sistemático y que no son explicadas por las otras tres componentes

Existen diferentes modelos de combinación de las componentes. Para describir los modelos necesitamos primero una nomenclatura básica. Denotando por X_t al valor de la variable en el instante t , se tiene:

$$X_t = f(T_t, E_t, I_t)$$

donde:

- T_t : Valor de la tendencia en el instante t
- E_t : Valor de la componente estacional en el instante t
- I_t : Valor de la componente irregular en el instante t (ruido).

Por tanto, los modelos que puede adoptar la función f son los siguientes:

- **Modelo multiplicativo:** La composición de la serie se realiza mediante el producto de sus componentes.

$$X_t = T_x \times E_t \times I_t$$

- **Modelo aditivo:** Las componentes se agregan para formar la serie temporal.

$$X_t = T_x + E_t + I_t$$

- **Modelo mixto:** La composición de la serie de la parte irregular viene de forma aditiva y la parte regular de forma multiplicativa.

$$X_t = T_x \times E_t + I_t$$

Tras haber detectado el modelo mas adecuado, podremos conocer el comportamiento de la serie a largo plazo.

El siguiente paso realizar una estimación de la tendencia, T_t , habiendo eliminado previamente la componente estacional para impedir que estas oscilaciones perturben la identificación de la tendencia.

Para estimar T_t , debemos hacer una hipótesis sobre su forma:

- **Tendencia determinista:** Se supone que la tendencia es una función determinística del tiempo:

$$T_t = a + bt \quad a, b \in \mathbb{R}$$

Siendo a y b constantes, que se estimarán mediante un modelo de regresión lineal.

Sin embargo, el método que aplicaremos será el que exponemos a continuación:

- **Tendencia evolutiva (método de medias móviles):** Este método consiste en definir la tendencia como una serie suavizada. Supondremos que la tendencia de la serie es una función que evoluciona lentamente y que podremos aproximar función simple del tiempo, suponiendo así una recta.

Una vez identificada la tendencia, procedemos a hacer un análisis de la estacionalidad de la serie, con el objetivo de:

- **Desestacionalizar la serie**, es decir, eliminar las oscilaciones periódicas que se repiten a lo largo de los años, haciendo así que los datos de distintas estaciones sean comparables. La serie desestacionalizada la conseguimos diferenciando la serie.
- **Realizar predicciones**, ya que si nuestros datos están afectados por una componente estacional, necesitaremos una estimación de esta de cara a realizar una predicción

Para desestacionalizar la serie, emplearemos los índices de variación estacional asociados a cada estación, ya que se suponen constantes año a año. Con esta técnica, se evidencian las diferencias en cada período, por ejemplo, podemos ver la diferencia del volumen de ventas en función de la época del año (mes, día de la semana, estación,...) Estos índices reflejan la cantidad fija o proporción en la que se modifica la tendencia en cada estación.

Una vez calculados estos índices, se desestacionaliza la serie, eliminando así el efecto de cada estación.

Por último, procedemos a realizar las predicciones. Para ello, necesitamos que se cumpla la condición de estacionariedad, es decir, la media y la varianza permanecen constantes en el tiempo (no tiene raíces unitarias). En el caso de no imponer esta condición de estacionariedad, predeciríamos características que no serán las mismas en el futuro que en el pasado.

Se tiene que todo proceso lineal es estacionario, por tanto, obtendremos trabajaremos con series estacionarias, y de lo contrario, podemos aplicar los mismos métodos a series no estacionarias realizando las transformaciones pertinentes para conseguir la estacionariedad.

En nuestro caso, aplicaremos la metodología Box-Jenkis como método predictivo.

3.2.1. Metodología Box-Jenkis

Esta metodología tiene en cuenta la dependencia existente entre los datos, es decir, cada observación en el instante t será modelada a partir de los valores pasados. Los modelos se conocen con el nombre de ARIMA (modelos integrados autorregresivos de medias móviles), que deriva de las siguientes componentes: AR (Autorregresivo) , I (integrado), MA(Medias móviles)

El siguiente paso es identificar el modelo más adecuado a través del estudio de la función de autocorrelación (FAC) y la función de autocorrelación parcial (FAP).

Nota: el método recomienda como mínimo 50 observaciones en la serie temporal.

Fases de la metodología Box-Jenkis:

1. Identificar el la estructura ARIMA que sigue la serie a través del estudio de la función de autocorrelación simple (FAS) y la función de autocorrelación parcial (FAP). Determinar el modelo arima consiste en identidicar los órdenes p y q de su estructura autoregresiva y de medias móviles

2. Estiamción de parámetros: Una vez tenemos identificado el modelo, estimamos los parámetros AR y MA del modelo por el método de máxima verosimilitud, obteniendo el error estándar y los resíduos del modelo

Nota: Es muy importante comprobar que las estimaciones son significativamente no nulas.

3. Diagnosis del modelo: Comprobamos que los resíduos sigan un proceso de ruido blanco mediante el Test de Ljung-Box.

Si hemos identificado varios modelos y todos ellos pasan la diagnosis, nos quedaremos con uno de ellos según el criterio del menor AIC

4. Predicción: una vez identificado y validado el mejor modelo, se realizan las predicciones con éste.

Capítulo 4

Proceso de la ciencia de datos (Data science process)

4.1. Introducción

La ciencia de datos es la combinación de múltiples campos, como la estadística, la inteligencia artificial (IA), el análisis de datos,... con el objetivo de extraer información de valor de los datos. La ciencia de datos, abarca las siguientes etapas: recolección de los datos, limpieza, análisis exploratorio, construcción y validación de modelos y predicciones.

Una parte importante de la ciencia de datos es el Aprendizaje Automático o Machine Learning (ML). Se trata de un subcampo dentro de la ciencia de datos, concretamente, una subcategoría de la inteligencia artificial. Está basada en algoritmos, y consiste en que éstos descubran de manera autónoma patrones recurrentes del conjunto de datos. Los algoritmos de ML al detectar patrones en los datos, aprenden y mejoran el rendimiento en la ejecución de una tarea o al hacer predicciones. Una vez entrenado y validado el modelo, el algoritmo podrá encontrar patrones en nuevos datos (predicciones)

Para la correcta explicación de las técnicas que se van a describir, es necesario la definición del aprendizaje estadístico supervisado.

El aprendizaje estadístico supervisado es una de las principales herramientas del aprendizaje automático y consiste en una serie de técnicas para deducir una función a partir de una serie de datos de entrenamiento. El objetivo es crear o estimar una función capaz de predecir el valor deseado después de haber visto una serie de ejemplos. Para ello, tiene que generalizar a partir de los datos presentados anteriormente a las nuevas situaciones no vistas previamente. La salida de la función puede ser un valor numérico (como en problemas de regresión) o una etiqueta de clase (como en los de clasificación).

4.2. Etapas del proceso de la ciencia de datos

A continuación se van a exponer las diferentes etapas que es necesario completar para la correcta realización de un proceso de ciencia de datos (DSP; Data Science Process)

4.2.1. Conocimiento del negocio (Knowledge of Business)

En esta primera etapa, es fundamental la definición del problema que nos ocupa, la definición de unos objetivos claros y la metodología para cumplirlos.

Esto implica la comprensión de los requisitos del proyecto desde el punto de vista de negocio, utilizando las perspectivas de negocio para determinar a qué problemas podemos dar respuesta mediante el uso de la minería de datos.

4.2.2. Adquisición de los datos (Collect the data)

Explicación de los datos, fuente, explicación de las variables,...

Consiste en explicar como se ha llevado a cabo la adquisición de los datos, la identificación de las distintas fuentes y la explicación de los mismos.

4.2.3. Preparación de los datos (data preparation)

Raramente encontraremos los datos preparados para su análisis, ya que normalmente es necesario la limpieza y la transformación de los mismos. Para ello, es necesario llevar a cabo un paso previo llamado pre-procesamiento de los datos.

Fases de la preparación de los datos (data cleaning):

- Eliminación de duplicados (filas y columnas)
- Datos erróneos (ej: precios negativos)
- Detección de valores faltantes: decidir si eliminar esos registros o imputarlos
- Detección de outliers (decidir si mantener, quitar o tratar a parte)
- Unificación de variables (unificación de unidades,...)
- Creación de variables a partir de otras ya existentes si fuera necesario

Preparación de los datos

- Reformateo de variables, por ejemplo, formatos horarios.
- Categorización,...
- Selección de variables (Feature selection): elegir las mejores variables que alimenten nuestros algoritmos dictarán la máxima calidad que podemos conseguir, ya que no todas las variables explican el problema que queremos modelar. Podemos resumir esto con la siguiente frase: “Garbage in, garbage out”, es decir, si entra basura saldrá basura. Refiriéndonos con basura a ruido en los datos o información pobre.

4.2.4. Análisis exploratorio de datos (EDA)

El análisis exploratorio se utiliza para ver lo que nos pueden ofrecer los datos antes de la etapa del modelado y se lleva a cabo para resumir las principales características del conjunto de datos a través de diferentes tareas:

- Estudio descriptivo de los datos: La estadística descriptiva es la parte de la estadística dedicada a la ordenación y tratamiento de la información por medio de gráficas y tablas, además de la obtención de parámetros útiles para explicar la información
- Visualizaciones de los datos:



Figura 4.1: División de datos muestrales para entrenamiento, validación y testeo

- Análisis univariante: Empleado para observar diferentes características de interés, tratar de identificar patrones en los datos o ver la distribución de las variables . Algunos ejemplos serían los gráficos de caja y bigote o histogramas
- Análisis multivariante: Donde tratamos de ver la asociación o relación que pueden tener las distintas variables de interés. Encontramos los gráficos de barras o gráficos de dispersión entre los ejemplos de representaciones multivariantes.
- Relación entre las variables

Este tipo de análisis permite obtener medidas descriptivas de un conjunto de datos para poder extraer conclusiones referentes a una muestra o población.

4.2.5. Modelado

En la etapa de modelado aplicaremos algoritmos de aprendizaje automático. Para llevar a cabo esta fase y con el objetivo de obtener mayor robustez en los modelos, aplicaremos la técnica conocida como *hold out*.

El *hold out* es una técnica en la que dividimos los datos en dos partes mutuamente excluyentes (no superpuestas), utilizando una de ellas para el entrenamiento de los modelos y la otra para el testeo.

La traducción literal para hold-out es *retención* y esta técnica recibe este nombre porque reservamos una parte de los datos para probar el modelo en datos nuevos.

Esta técnica se emplea para evitar el sobreajuste. Este aparece cuando un modelo que se adapta perfectamente a los datos de entrenamiento obteniendo unas métricas muy buenas pero que luego es incapaz de generalizar con datos nuevos, y por tanto, existe una sobrevaloración de la capacidad predictiva de los modelos obtenidos.

Por tanto, dividimos los datos en el conjunto de datos de entrenamiento, validación y testeo. Para ello, generamos un conjunto de entrenamiento y otro de testeo a partir del conjunto de datos muestral. A continuación, volvemos a dividir los datos de entrenamiento en datos de entrenamiento y validación, obteniendo así tres conjuntos de datos: entrenamiento, validación y testeo.

El conjunto de *datos de entrenamiento* es aquel que utilizamos para probar diferentes hiperparametrizaciones de cada modelo para ver cual es la más óptima. La hiperparametrización variará en función de los parámetros aplicables a cada algoritmo utilizado.

Una vez hayamos entrenado los modelos, pasamos a la fase de validación, donde aplicaremos a los *datos de validación* los diferentes algoritmos con la configuración de parámetros que mejor haya funcionado en el conjunto de datos de entrenamiento.

El modelo con el que obtengamos las mejores métricas será el que posteriormente apliquemos a los *datos de testeo*, ofreciéndonos el error real cometido con el modelo seleccionado. Es decir, este último conjunto de datos se utiliza para estimar el error de generalización del modelo, ya que nuestro objetivo es obtener un error de generalización pequeño evitando el sobreajuste.

A continuación vamos a exponer tres algoritmos de aprendizaje automático que posteriormente aplicaremos a nuestros datos de ventas de productos.

4.2.5.1. Máquinas de vector soporte (Support Vector Machines SVMs)

Las máquinas de vector soporte son un conjunto de algoritmos de aprendizaje estadístico supervisado pertenecientes a la familia de los clasificadores lineales. Este algoritmo, más conocido como SVM fue desarrollado en los laboratorios AT&T Bell por Vapnik y otros autores a mediados del 1960, inicialmente para problemas de clasificación binaria, basados en la idea de separar los datos mediante hiperplanos. Actualmente existen extensiones dentro de esta metodología para clasificación con más de dos categorías, para regresión y también para la detección de datos atípicos. La idea fundamental es la utilización de vectores que hacen de soporte con el fin de maximizar la separación entre los datos y el hiperplano.

Suponiendo que tenemos ejemplos de sólo dos categorías y sin pérdida de generalidad, una SVM construye un hiperplano en un espacio de dimensionalidad muy alta. Este hiperplano separa de forma óptima los puntos de una clase de la otra. La característica fundamental de estos algoritmos es el concepto de “separación óptima”, ya que se busca el hiperplano que tenga la máxima distancia con los puntos que estén más cerca de él mismo al tiempo que clasifica correctamente tantos puntos de entrenamiento como sea posible. Los algoritmos SVM representan el hiperplano óptimo con vectores de soporte.

En nuestro caso al ser la variable volumen de ventas una variable numérica, vamos a centrarnos en la variante SVM para regresión, también conocida como SVR (support vector regressor). El caso del problema de regresión es una generalización del problema de clasificación, en la que el modelo devuelve un valor continuo, es decir, un modelo de regresión estima una función multivariante de valor continuo.

4.2.5.1.1. Descripción del algoritmo

Dado un conjunto de ejemplos de entrenamiento $S = \{(x_1, y_1), \dots, (x_n, y_n)\}$, donde $x_i \in \mathbb{R}^d$ e $y_i \in \mathbb{R}$, en el que se asume que todos los valores y_i de todos los ejemplos de S pueden ser ajustados mediante un hiperplano, nuestro objetivo será encontrar los parámetros $w = (w_1, \dots, w_d)$ que permitan definir el hiperplano de regresión

$$y = f(x) = (w_1x_1 + \dots + w_dx_d) + b = \langle w, x \rangle + b, b \in \mathbb{R}$$

La generalización de SVM a SVR se logra introduciendo una región insensible a ϵ alrededor de la función. Esta región se conoce como tubo épsilon. Este tubo reformula el problema de optimización para encontrar el tubo que mejor se aproxime a la función al tiempo que equilibra el error de predicción, es decir, se formula un problema de optimización definiendo una función de pérdida a minimizar insensible a ϵ y encontrando el tubo más plano que contiene a la mayoría de instancias de entrenamiento.

Se dice **rudio, perturbación aleatoria o tubo épsilon** y se representa por $\epsilon \sim N(0, \sigma^2)$, al error en la medición del valor y , por tanto, $y = f(x) + \epsilon$

El valor de ϵ determina el ancho del tubo, y un valor más pequeño indica menor tolerancia al error, cuando más pequeño sea el valor de ϵ , el límite del tubo se desplaza hacia dentro, habiendo más puntos de datos alrededor del límite, lo que indica más vectores de soporte.

Se define la **función de pérdida lineal ϵ -insensible**, y se representa como L_ϵ a una función lineal en el que la función de pérdida toma valor nulo y viene definida de la siguiente forma:

$$L_\epsilon = \begin{cases} 0 & \text{si } |y - f(x)| \leq \epsilon, \\ |y - f(x)| - \epsilon & \text{en caso contrario} \end{cases}$$

Por tanto, el problema fué planteado por Vapnik como el siguiente problema de optimización:

$$\text{Min}_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n (\xi_i + \xi_i^*)$$

$$s.a = \begin{cases} y_i - w \cdot x_i - b \leq \epsilon, +\xi_i \\ w \cdot x_i + b - y_i \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \quad \forall i \end{cases}$$

Cuando el error es menor que ϵ , las variables de holgura valen 0. Para resolverlo, podemos recurrir al problema dual y al uso de funciones base para trabajar en espacios de mayor dimensión.

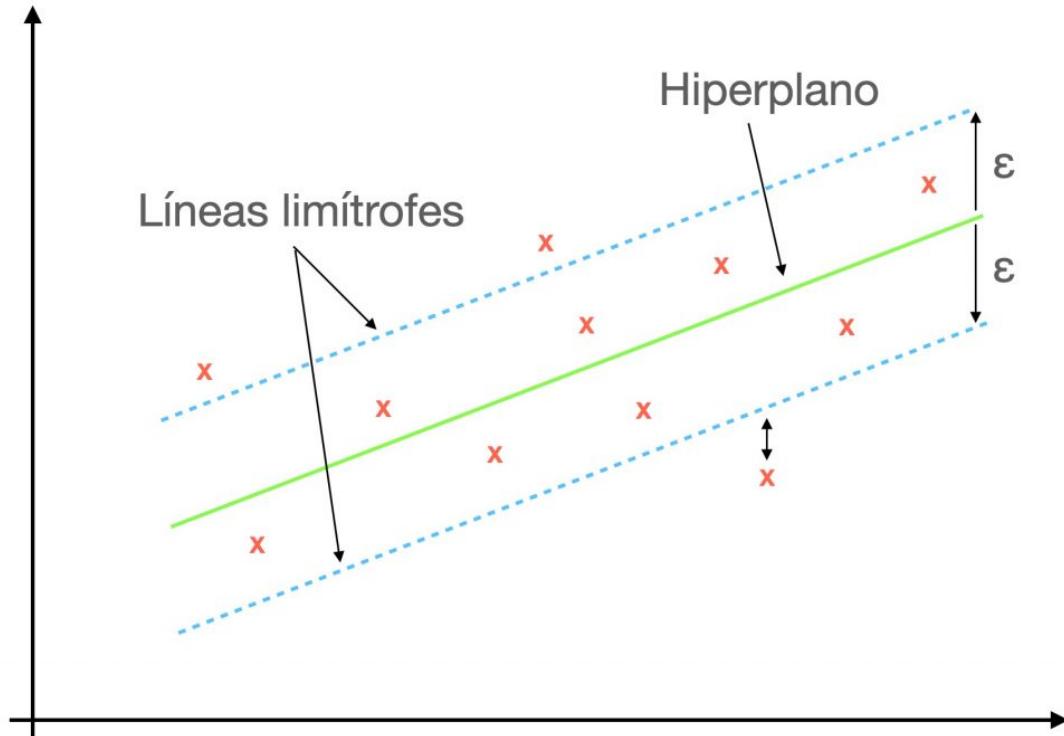


Figura 4.2: Vectores de Soporte de Regresión.

4.2.5.2. K-Nearest Neighbor Regression (KNN)

El algoritmo de K-vecinos más cercanos, más conocido como KNN, fué desarrollado en el año 1951 por los matemáticos Evelyn Fix y Andrew Hodges.

El algoritmo KNN es un método de aprendizaje supervisado que está basado en criterios de vecindad, por lo que es necesario establecer cierta medida de distancia entre los diferentes elementos de la representación. La ventaja de la aplicación de técnicas basadas en la vecindad es la siguiente: el valor de salida que se otorgará a una nueva instancia se calculará en función de los valores de los puntos más cercanos a ella. Se trata de un método local, que asume que la salida de un nuevo dato depende exclusivamente de los k vecinos de entrenamiento más próximos.

4.2.5.2.1. Descripción del algoritmo

Este algoritmo puede ser utilizado para modelos de clasificación y de regresión, ocupándose en este trabajo la segunda opción. En el caso de la clasificación, se determinará la clase a la que pertenecerá la nueva instancia en función de la clase mayoritaria de los vecinos más cercanos del conjunto de entrenamiento; y en regresión, el modelo debe determinar el valor del nuevo dato como el valor medio de los k ejemplos de entrenamiento más cercanos, siguiendo la siguiente ecuación del valor de la nueva instancia de entrada:

$$Valor(Inst_{\text{entrada}}) = \frac{1}{K} \sum_{i=1}^k Valor(P_i)$$

Como ya habíamos avanzado antes, para determinar como de cercanos se encuentran unas instancias de otras, es necesario definir una medida de similitud o distancia para todos los datos del conjunto muestral. Definiremos esta medida de similitud a través de una función, como puede ser la distancia Manhattan, la distancia Minkow o las más utilizada, la distancia Euclídea, que es la que se va a utilizar, y viene dada por:

$$d(p, 1) = \sqrt{\sum_{i=1}^n (p_i - q_i)^2}$$

Una vez definida esta medida, procedemos a la descripción del algoritmo:

- Se almacena el conjunto de datos de entrenamiento compuesto por un vector de entrada y otro de salida
- Se establece el valor del parámetro k
- Se presenta una nueva instancia j teniendo en cuenta únicamente el vector de entrada de esta nueva instancia
 - Se calcula la distancia euclídea de la nueva instancia con todos los datos del conjunto de entrenamiento
 - Se calcula la salida del este nuevo dato como la media de las salidas de los k datos más cercanos a él
- Se repite el paso anterior para todas las instancias del conjunto de datos

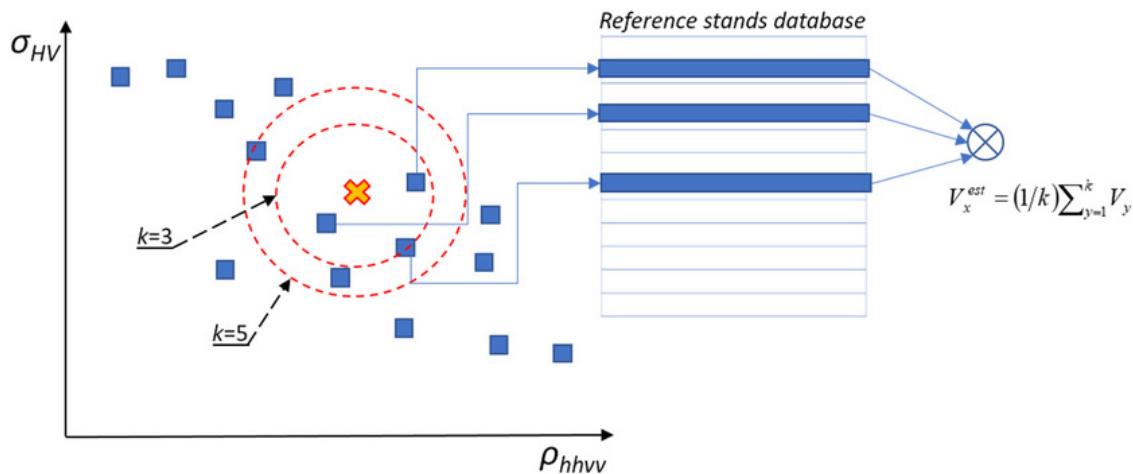


Figura 4.3: Esquema conceptual del algoritmo KNN de regresión.

En la figura 4.3 podemos ver un esquema conceptual del algoritmo, donde el valor de salida que el modelo le dará al nuevo punto marcado con una x será la media de los valores de los puntos vecinos. Habiéndose seleccionado a modo de ejemplo el valor de $k=3$ y $k=5$.

4.2.5.2.2. Elección del parámetro k

Es necesario seleccionar el valor que se le va a dar al parámetro k , es decir, el número de vecinos con los que se realizará la media para obtener el valor de salida de la nueva instancia. Si este valor es muy grande, la idea de vecinos que están lejos podrían influir con

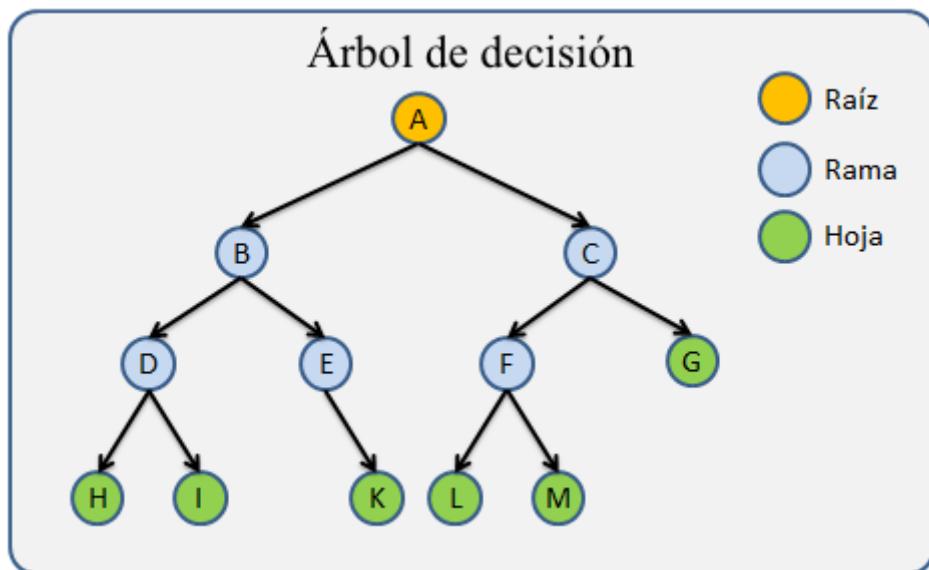


Figura 4.4: Árbol de decisión. Partes

la nueva instancia sin tener relación, pero si este valor es demasiado pequeño, el algoritmo será muy sensible a valores extremos.

4.2.5.3. Árboles de decisión (XGBoost Model)

Los árboles de regresión conducen a dividir o segmentar el espacio predictor en regiones más simples de tal forma que la predicción de una instancia se hará a través de la media (o moda) de la región a la que pertenece.

En el caso de los árboles de decisión, el conjunto de reglas empleadas para la segmentación del espacio de predicción se puede resumir en un árbol.

El análisis de árboles de clasificación y regresión, generalmente consiste en tres fases:

- **Construcción del árbol máximo:** este árbol se construye empleando un procedimiento de partición binaria, comenzando en la raíz del árbol. Este primer árbol describe el conjunto de entrenamiento y contiene gran cantidad de niveles (sobreajuste) y nodos, pudiendo ser demasiado complejo. Cada grupo es categorizado por la media (regresión) o la distribución (clasificación) de la variable respuesta, el tamaño del nodo y los valores de las variables explicativas que lo definen.

En la figura anterior, podemos ver la representación de un árbol de decisión desde el nodo raíz. Entre sus componentes encontramos: *las ramas*, que son los segmentos del árbol que conectan los nodos; *los nodos internos*, puntos a lo largo del árbol donde se va dividiendo el espacio predictor y *los hojas* (nodos terminales).

- **Poda del árbol:** El árbol máximo está generalmente sobreajustado y por tanto, un árbol más pequeño con menos divisiones podría conducir a una menor varianza. Por este motivo, procede a la poda de éste cortando ramas hasta encontrar el tamaño “adecuado” del árbol.

Una forma de resolver el problema es generar una serie de árboles anidados (árboles de secuencia anidada) de tamaños decrecientes, seleccionando para cada tamaño el mejor de

todos. Posteriormente, se comparan para determinar el óptimo mediante el criterio de coste-complejidad.

Para cada árbol T , se define la función de costo-complejidad, y se representa como $R_\alpha(T)$, como: $R_\alpha(T) = R(T) + \alpha|\tilde{T}|$,

donde

- $R(T)$ indica el promedio de la suma de cuadrados entre los nodos
- $|\tilde{T}|$ indica la complejidad del árbol, que se define como el número total de nodos terminales
- α es el parámetro de complejidad, valores altos de este parámetro indican árboles pequeños
- **Selección del árbol óptimo mediante validación cruzada:** El objetivo es seleccionar uno de los árboles de todos los árboles podados como el árbol óptimo, que será el árbol solución. El método de selección consistirá en asociar una medida de error a cada árbol y elegir el que tenga asociado un menor error.

El parámetro de complejidad definido es el que controla la compensación entre la complejidad (tamaño) del árbol y el ajuste a los datos de entrenamiento. Cuando $\alpha = 0$, el subárbol T es el árbol máximo. Sin embargo, a medida que aumenta α , las ramas del árbol se podan de forma anidada, siendo sencillo conseguir una secuencia completa de subárboles en función del valor del parámetro de complejidad. Podemos seleccionar un valor de α a través de un conjunto de validación cruzada.

Para la descripción del algoritmo *Extreme Gradient Boosting*, también conocido como XGBoost, es necesario la definición de varios conceptos, los *bosques aleatorios* y el método de aprendizaje estadístico *Boosting*.

Los bosques aleatorios (*Random Forest*) son una extensión de los árboles de clasificación. El modelo Random Forest es una técnica utilizada tanto para clasificación como para regresión basada en un conjunto de árboles de decisión. Este método selecciona submuestras del conjunto de datos inicial, asegurando así el uso de todas las variables y datos para construir el modelo, haciéndolo además idóneo para trabajar con grandes conjuntos de datos.

El método *boosting* tiene como propósito la reducción del sesgo. Se trata de un proceso iterativo que en lugar de ajustar un árbol de decisión, aplica la técnicas repetidas veces de forma secuencial, y por tanto, el algoritmo va aprendiendo lentamente. Este método no se aplica sobre los datos, sino sobre los resíduos.

Dado un árbol que ha sido previamente ajustado, se aplica un nuevo árbol para los resíduos del modelo, permitiendo así el reajuste del modelo. Este nuevo árbol de decisión construido con los resíduos se añade dentro de la función ajustada con el fin de mejorar el algoritmo en cada iteración y no de forma global.

Para aplicar esta técnica, es necesario fijar una serie de parámetros:

- Tamaño del árbol **d**, es decir, el número de nodos terminales
- El número de árboles **B**. Nota: un valor muy alto podría llevar a sobreajuste
- El parámetro de regularización λ , que puede ser interpretado como una proporción de aprendizaje, es decir, la velocidad a la que aprende el algoritmo. Se trata de un parámetro acotado entre 0 y 1, siendo habitual elegir $\lambda = 0.01$ o $\lambda = 0.001$.

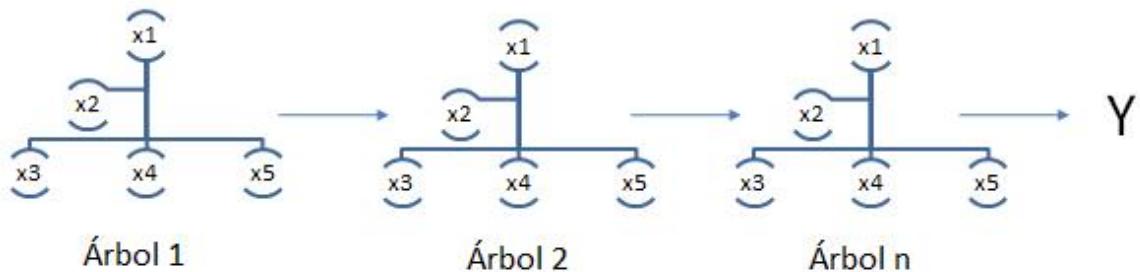


Figura 4.5: Algoritmo XGBoost

Este tipo de técnicas es aplicable tanto a problemas de regresión como de clasificación, sin embargo, nosotros nos centraremos en las de regresión, especialmente en el modelo *Extreme Gradient Boosting Algorithm*, tambien conocido como XGBoost.

Este algoritmo se encuentra dentro del marco de algoritmos de aprendizaje supervisado, y fué propuesto en el año 2016 por Chen y Guestrin y presenta las siguientes características:

- Consiste en la agregación de árboles de manera secuencial con el objetivo de aprender el resultado de los árboles previos y corregir el error producido por éstos, hasta que no se pueda reducir más el error (gradiente descendente)
- Para evitar el sobreajuste, realia un procesamiento en paralelo, la poda de árboles, el control de los valores perdidos y la optimización que penaliza la complejidad de los modelos.

4.2.5.3.1. Descripción del algoritmo

EL funcionamiento del algoritmo se puede resumir en cuatro pasos:

Dada una muestra inicial de aprendizaje $\{(x_i, y_i) : i = 1, \dots, n\}$.

- En primer lugar, se obtiene un árbol inicial con d divisiones, T_o para predecir la variable respuesta Y , asociando el resultado a un valor residual r_i
- Se obtiene un nuevo árbol R que se ajusta al error del paso anterior
- Los resultados de los árboles T_o y R se combinan para obtener un árbol T_1 , donde el error cuadrático medio será menor que el del árbol inicial
- Se continúa el proceso de forma iterativa hasta que el error es minimizado lo máximo posible

4.2.5.4. Evaluación y presentación de resultados (+análisis del error)

- Predicciones con el mejor modelo
- Final de la historia de una forma ordenada y resumida
- Señalar posibles mejoras y recomendaciones para proyectos futuros

Capítulo 5

Caso práctico con datos reales

En este apartado se va a llevar a cabo la aplicación de las diferentes técnicas estudiadas de forma teórica en los cuatro primeros capítulos.

En primer lugar, un estudio de cesta de la compra para descubrir asociaciones entre productos. Posteriormente, se llevará a cabo un DSP completo, que consta de las siguientes partes: extracción y lectura de datos, preprocesado, análisis exploratorio y modelado. En la parte de modelado, se aplicarán las diferentes técnicas estudiadas en la parte teórica: una serie de técnicas estadísticas clásicas como el modelo de regresión lineal y un análisis de series temporales, así como diferentes algoritmos de aprendizaje automático. El proceso de la ciencia de datos concluirá con la comparación del rendimiento de los diferentes modelos entrenados, el cálculo de errores y extracción de conclusiones.

Los datos que se han utilizado corresponden a una muestra de una base de datos anonimizadas empleadas en el proyecto *Advanced Promotional Engine*, dirigido por Dº Jose Luis Pino, tutor del TFG y se encuentran en formato DataFrame. La primera muestra de tickets, utilizada en el análisis de cesta de la compra contiene información correspondiente a transacciones de una cadena de supermercados. La segunda muestra, empleada para el proceso de ciencia de datos, corresponde a las ventas de dos productos lácteos similares de marcas distintas en un período de cinco meses.

5.1. Análisis de cesta de la compra para productos lácteos

5.1.1. Introducción

El objetivo principal de este apartado es aplicar un análisis de cesta de la compra para un conjunto de datos con una muestra de 7801 tickets que incluyen 4631 artículos distintos.

Aplicaremos esta técnica para descubrir los patrones de compra de los clientes y tratar así de identificar las relaciones existentes entre productos a la hora de comprar.

Para llevar a cabo este estudio haremos uso de la librería *arules*, un entorno creado para la identificación de reglas de asociación y conjuntos de ítems frecuentes.

5.1.2. Lectura y descripción de los datos

Los datos corresponden a una muestra de una base de datos anonimizadas empleadas en el proyecto *Advanced Promotional Engine*, dirigido por el tutor del TFG y se encuentran en formato *dataFrame*. Contienen información correspondiente a transacciones de una muestra de tickets de una cadena de supermercados.

Cada fila de nuestro conjunto de datos corresponde a una línea de un ticket y por tanto, una fila se corresponde con la venta de un determinado producto. Por este motivo, para una única transacción encontraremos tantas filas como productos diferentes se hayan comprado, y también encontramos registrada la cantidad de unidades comprada de cada producto.

En este conjunto de datos inicial encontramos las siguientes variables:

- **Idticket:** Variable numérica que identifica únicamente a cada ticket
- **Línea:** Variable numérica con la línea correspondiente del ticket
- **Item:** Item concreto
- **Cantidad:** Se trata de la cantidad de unidades que se ha comprado de un determinado ítem en una transacción concreta

A continuación mostramos brevemente algunas de las filas de nuestros datos:

Idticket	línea	item	cantidad
1	1	11802	12
1	2	21662	12
1	3	27959	3

Sin embargo, para aplicar un análisis de cesta de la compra necesitamos que nuestros datos estén en *formato cesta* (*formato basket*). Para obtener este formato, es necesario que cada línea del nuevo conjunto de datos corresponda a una única transacción, es decir, que en cada fila estén contenidos todos los productos que se refieren a sola compra.

Por ello, únicamente necesitamos una columna en la que tengamos recogidos todos los ítems pertenecientes a cada transacción, por lo que tendremos así tantas filas como transacciones, es decir, tantas filas como tickets generados.

Existen un total de 4631 productos y 7801 transacciones.

Procedemos a transformar el conjunto de datos inicial en uno nuevo para poder aplicarle funciones de la librería *arules*.

A continuación vamos a ver las seis últimas filas de nuestro nuevo conjunto de datos.

```
TicketsAgrupados<-transacciones %>% group_by(Idticket) %>% select(item, linea)

# Con el siguiente comando, conseguimos agrupar en una misma columna
# todos los ítems que corresponden a una misma transacción
DatosBasket <- ddply(TicketsAgrupados, c("Idticket"),
                      function(TicketsAgrupados)
                        paste(TicketsAgrupados$item, collapse = ","))
```

	Idticket	Items
7452	7508	11865,11865,26935
7162	7217	26545,22324,14654,25509,22541
7269	7324	24147,28212,22350
1004	1013	12033,34810,33229
623	628	27942,19195
7049	7103	17216,26129,27379,27382,1134,12337,18900,19156,19155,20118,1046,19626,1084,1033
2693	2709	23346,25825,19196,61957,13433,16460
934	941	26131,26131,21713,21713,17224,17224
4496	4523	25072,18962,18960,25038,10026,28863,28863,10026,24231,20484,26229,10026
2948	2968	19200,10566

```
# DatosBasket1 <- DatosBasket %>%
# rename("Transaccion" = V1) # Me daba error al compilar

DatosBasket1 <- DatosBasket
names(DatosBasket1) <- c("Idticket", "Items")

set.seed(1234)
DatosBasket1[sample(nrow(DatosBasket1), size = 10), ] %>%
  kable(booktabs=TRUE) %>%
  kable_styling(latex_options = c("striped", "scale_down"))
```

Observación: Es necesario mencionar que en cada transacción vemos una cota inferior de los items que han sido comprados, por ejemplo, vemos que en la transacción 7796 se han comprado los productos 15457 y 26978, pero no hemos considerado cuántos items de cada producto pertenecieron a esta compra. Esto se debe a que posteriormente al hacer uso de la función *read.transactions*, esta no va a considerar el número de items de cada producto que han sido comprados, sino que únicamente va a utilizar el que hayan sido o no comprados.

La librería arules contiene una serie de funciones para poder encontrar reglas de asociación entre productos, y por este motivo, a pesar de tener información sobre el número de items que son comprados de cada producto para una misma transacción, esta información no nos va a ser de utilidad a la hora de analizar las relaciones entre los productos.

Por último, después de haber transformado nuestros datos, guardamos la columna correspondiente a los productos en un archivo *.csv* para poder posteriormente leerlo correctamente haciendo uso de la función *read.transactions* perteneciente a la librería *arules*.

5.1.3. Análisis de ventas

Leemos los datos y vemos una primera información a modo resumen de éstos:

```
## transactions as itemMatrix in sparse format with
## 7801 rows (elements/itemsets/transactions) and
## 4631 columns (items) and a density of 0.001169421
```

```
##  
## most frequent items:  
##    1033    28716    1096    26785    24347 (Other)  
##      507      451      358      294      266    40371  
##  
## element (itemset/transaction) length distribution:  
## sizes  
##    1     2     3     4     5     6     7     8     9     10    11    12    13    14    15    16  
## 1847 1285 952 708 552 389 336 277 221 177 153 109 98 102 70 64  
##    17    18    19    20    21    22    23    24    25    26    27    28    29    30    31    32  
##    71    50    50    40    34    22    32    20    21    13    14    12     9     9     8     6  
##    33    34    35    36    37    38    39    40    41    42    43    44    45    48    49    50  
##    5     1     4     8     3     2     3     4     1     1     2     2     1     1     1     2  
##    52    54    55    56    59    60    72    82  
##    1     1     2     1     1     1     1     1  
##  
##      Min. 1st Qu. Median      Mean 3rd Qu.      Max.  
##      1.000   2.000   3.000   5.416   7.000  82.000  
##  
## includes extended item information - examples:  
## labels  
## 1 100  
## 2 10002  
## 3 10004
```

Observamos que hay un total de 7801 transacciones y 4631 artículos vendidos. Las transacciones son los subconjuntos de estos 4631 artículos.

En el resumen de los datos podemos ver otra información que nos puede ser útil:

- Density: Se trata del número total de artículos que se han comprado entre el número total de artículos existentes, en nuestro caso: $\text{densidad}=0.001169$.
- Productos más frecuentes:

Tabla 5.1: Productos más frecuentes

ID Producto	Cota inferior de unidades vendidas
1033	507
28716	451
1096	358
26785	294
24347	266
Otro	40371

- La media de productos por transacción es de 5 artículos. Además, cabe destacar que en el 75 % de las transacciones se han comprado 7 artículos o menos.
- Tamaño de las transacciones: un total de 1847 transacciones fueron de un único artículo y 1285 transacciones fueron de dos artículos, indicando estos resultados que la mayoría de clientes compraron entre 1 y 2 artículos. La transacción con más

productos diferentes ha sido una transacción con 82 artículos.

Nota: Recordemos que el número de items es una cota inferior, es decir, para una transacción con un único artículo, sabemos que se compró al menos una vez, pero no sabemos cuantas unidades se compraron.

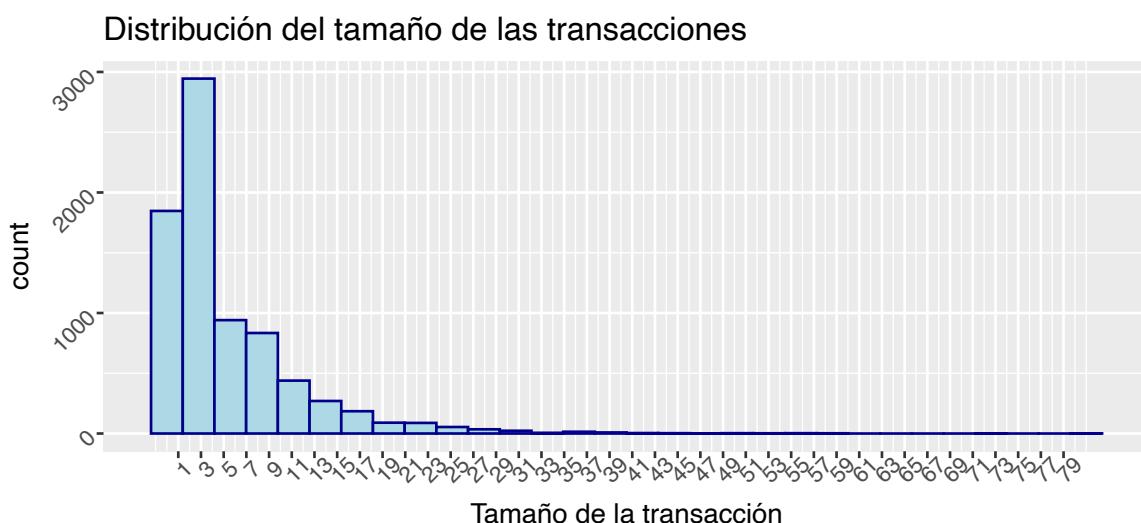
A continuación vamos a ver una lista con algunas transacciones:

```
## [1] "{16591,22852,22858,24232}" "{15005,20191,20676}"
## [3] "{24945,26737,33702,33753}" "{21134,21135}"
## [5] "{26042,26785}"           "{23570,24347}"
```

- Estudio de los cuantiles y la distribución del tamaño de las transacciones:

```
## 0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
## 1   1   1   2   2   3   4   6   8   13  82
```

Vamos a mostrar gráficamente la distribución de los tamaños de las transacciones:



Vemos que los productos 1033, 28716 y 1096 son los tres artículos más vendidos de entre todos los existentes.

También es importante estudiar como se distribuye el soporte de los productos individuales, para posteriormente establecer un límite de soporte. Esto se puede calcular fácilmente con un análisis de los ítems más frecuentes (con mayor soporte) dentro del conjunto de transacciones.

```
frec_items <- itemFrequency(x = TransBasket, type = "relative")
frec_items %>% sort(decreasing = TRUE) %>% head(5)

##      1033     28716      1096     26785     24347
## 0.06499167 0.05781310 0.04589155 0.03768748 0.03409819
```

En el listado anterior podemos observar que el 6.5% de las transacciones contiene al producto 1033, el 5.7% al producto 28716 y que en el 4.58% de éstas se ha vendido el producto 1096.

Vemos que el soporte individual de los ítems son bastante bajos, ya que tenemos un conjunto de datos con un gran número de transacciones y muchos productos diferentes.

Después de haber visto los aspectos más destacables de nuestros datos, procedemos a la aplicación del algoritmo a priori.

5.1.4. Aplicación del algoritmo a priori

Este algoritmo ya fué descrito en el desarrollo teórico, y nos permitirá generar una serie de reglas de asociación. Como hemos mencionado a lo largo de la descripción de este apartado práctico, el paquete *arules* también implementa el algoritmo *Apriori* para identificar ítemsets frecuentes y descubre reglas de asociación con la función *apriori*, donde indicaremos una serie de parámetros: soporte, confianza, tamaño mínimo o máximo y el tipo de asociación requerida (target)

5.1.4.1. Itemsets

En primer lugar, vamos a extraer ítemsets formados por al menos dos ítems que hayan sido comprado al menos 30 veces.

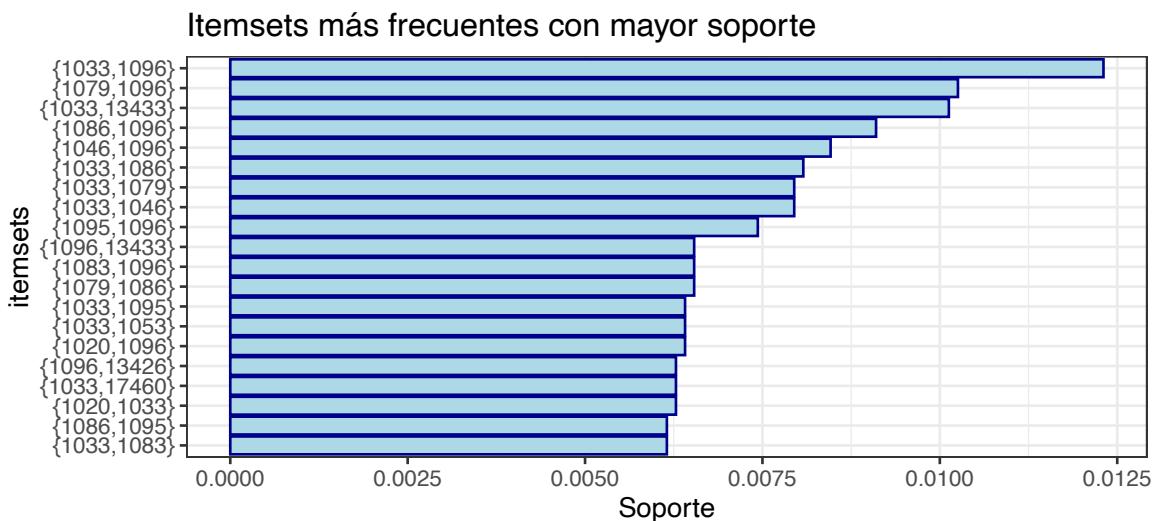
```
## Apriori
##
## Parameter specification:
##   confidence minval smax arem  aval originalSupport maxtime      support minlen
##             NA      0.1    1 none FALSE                  TRUE      5 0.003845661      2
##   maxlen      target  ext
##         80 frequent itemsets TRUE
##
## Algorithmic control:
##   filter tree heap memopt load sort verbose
##   0.1 TRUE TRUE FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 30
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[4631 item(s), 7801 transaction(s)] done [0.03s].
## sorting and recoding items ... [253 item(s)] done [0.00s].
```

```
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 done [0.00s].
## sorting transactions ... done [0.00s].
## writing ... [64 set(s)] done [0.00s].
## creating S4 object ... done [0.00s].

## set of 64 itemsets
##
## most frequent items:
##    1033    1096    1079    1086    1083 (Other)
##      25       15        9        9        8       63
##
## element (itemset/transaction) length distribution:sizes
##  2   3
## 63   1
##
##    Min. 1st Qu. Median     Mean 3rd Qu.     Max.
##    2.000 2.000 2.000 2.016 2.000 3.000
##
## summary of quality measures:
##      support      transIdenticalToItemsets      count
##    Min. :0.003846  Min. :0.0000000  Min. :30.00
##  1st Qu.:0.004198  1st Qu.:0.0000000  1st Qu.:32.75
##  Median :0.004871  Median :0.0000000  Median :38.00
##  Mean   :0.005482  Mean   :0.0001182  Mean   :42.77
##  3rd Qu.:0.006281  3rd Qu.:0.0001282  3rd Qu.:49.00
##  Max.   :0.012306  Max.   :0.0010255  Max.   :96.00
##
## includes transaction ID lists: FALSE
##
## mining info:
##      data ntransactions      support confidence
##  TransBasket          7801 0.003845661           1
```

Hemos encontrado un total de 64 itemsets con un soporte mayor al soporte mínimo indicado de 0.0038457. La mayoría de estos conjuntos de items están formados por dos items.

De estos itemsets encontrados, vamos a proceder a mostrar aquellos con mayor soporte:



Del gráfico anterior podemos observar que la dupla $\{1033, 1096\}$ tiene el mayor soporte, es decir, que se trata del itemset que más veces ha sido vendido. El soporte, 0.012306115 indica que estos productos han sido vendidos un 1.23 % del total de transacciones.

Vamos a filtrar los itemsets para seleccionar aquellos que contienen los productos 1033 y 1096.

```
##      items          support  transIdenticalToItemsets count
## [1] {1033,1096} 0.012306115 0.0003845661         96
## [2] {1033,1086,1096} 0.003845661 0.0000000000         30
```

La mayoría de veces, un total de 96, estos dos ítems han sido comprados exclusivamente, mientras que han sido comprado con otro producto, el 1086 hasta en 30 ocasiones.

5.1.4.2. Reglas de asociación

A continuación vamos a crear reglas de asociación de la misma forma que hemos identificado los itemsets, pero indicando un valor mínimo para el parámetro *confianza*, en este caso, del 50 %. Vamos a imponer la obtención de reglas cuyos itemsets hayan sido comprado al menos 20 veces.

```
##      lhs          rhs          support    confidence coverage     lift   count
## [1] {1084}      => {1083} 0.002563774 0.5000000 0.005127548 20.858289 20
## [2] {14240}     => {14655} 0.002820151 0.5000000 0.005640303 16.527542 22
## [3] {16037}     => {1096} 0.003076529 0.5000000 0.006153057 10.895251 24
## [4] {1033,13426} => {1096} 0.002563774 0.5128205 0.004999359 11.174617 20
## [5] {1020,1046}  => {1096} 0.002691963 0.6774194 0.003973850 14.761308 21
## [6] {1020,1096}  => {1033} 0.003332906 0.5200000 0.006409435 8.001026 26
## [7] {1020,1033}  => {1096} 0.003332906 0.5306122 0.006281246 11.562308 26
## [8] {1046,1086}  => {1096} 0.002563774 0.6896552 0.003717472 15.027933 20
## [9] {1046,1079}  => {1096} 0.002820151 0.6285714 0.004486604 13.696887 22
## [10] {1086,1095}  => {1096} 0.003076529 0.5000000 0.006153057 10.895251 24
## [11] {1079,1095}  => {1096} 0.002820151 0.5789474 0.004871170 12.615554 22
## [12] {1079,1086}  => {1096} 0.003461095 0.5294118 0.006537623 11.536149 27
## [13] {1079,1083}  => {1096} 0.002563774 0.5405405 0.004742982 11.778650 20
## set of 13 rules
```

```

## 
## rule length distribution (lhs + rhs):sizes
##  2 3
##  3 10
##
##      Min. 1st Qu. Median     Mean 3rd Qu.     Max.
##      2.000  3.000  3.000   2.769  3.000  3.000
##
## summary of quality measures:
##           support      confidence      coverage      lift
##      Min. :0.002564  Min. :0.5000  Min. :0.003717  Min. : 8.001
##  1st Qu.:0.002564  1st Qu.:0.5000  1st Qu.:0.004743  1st Qu.:11.175
##  Median :0.002820  Median :0.5294  Median :0.005128  Median :11.779
##  Mean   :0.002899  Mean   :0.5545  Mean   :0.005315  Mean   :13.025
##  3rd Qu.:0.003077  3rd Qu.:0.5789  3rd Qu.:0.006153  3rd Qu.:14.761
##  Max.   :0.003461  Max.   :0.6897  Max.   :0.006538  Max.   :20.858
##
##           count
##      Min. :20.00
##  1st Qu.:20.00
##  Median :22.00
##  Mean   :22.62
##  3rd Qu.:24.00
##  Max.   :27.00
##
## mining info:
##           data ntransactions      support confidence
## TransBasket          7801       0.002563774        0.5

```

Se han encontrado un total de 13 reglas. Sin embargo, el algoritmo nos está recomendando comprar los productos 1096 y 1033 en la mayoría de las reglas. Se trata de algunos de los productos más vendidos, por lo que no tiene sentido. Estos productos no tienen problemas para su venta, por lo que nuestro objetivo es buscar reglas que recomiendan productos que se han vendido en menor volumen.

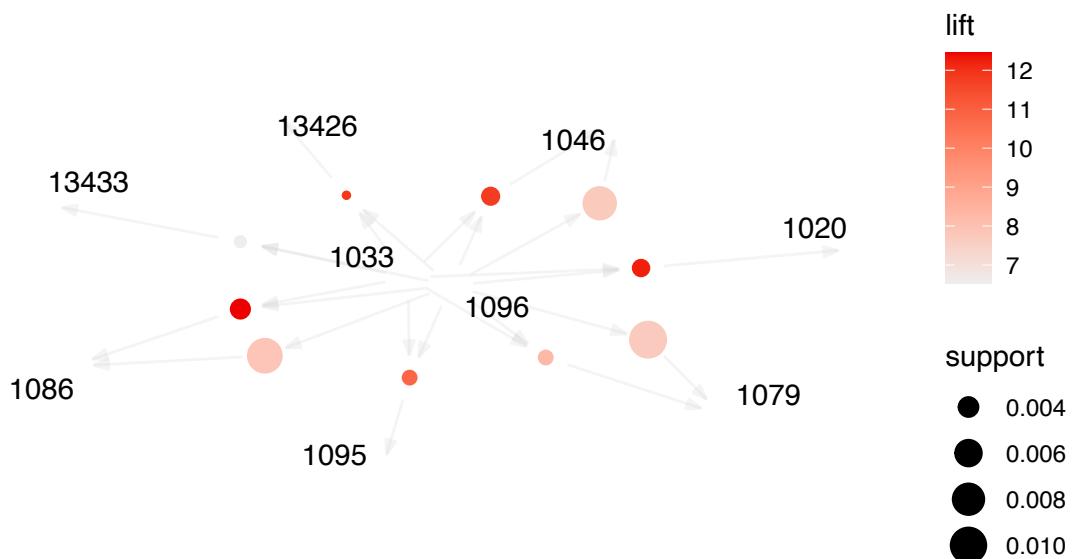
Para ello, vamos a modificar las reglas, bajando el valor de la confianza y obligando al algoritmo a tener los productos más frecuentes a la izquierda, en la parte de *lhs*.

Veamos las nuevas reglas de asociación:

	lhs	rhs	support	confidence	coverage	lift	count
## [1]	{1033,1096} => {1086}	0.003845661	0.3125000	0.01230611	12.437819	30	
## [2]	{1033,1096} => {1046}	0.003461095	0.2812500	0.01230611	11.795867	27	
## [3]	{1033,1096} => {1020}	0.003332906	0.2708333	0.01230611	12.212548	26	
## [4]	{1033,1096} => {1095}	0.002948340	0.2395833	0.01230611	10.803408	23	
## [5]	{1033,1096} => {1079}	0.002948340	0.2395833	0.01230611	8.269865	23	
## [6]	{1096} => {1079}	0.010255096	0.2234637	0.04589155	7.713452	80	
## [7]	{1033,1096} => {13433}	0.002691963	0.2187500	0.01230611	6.538194	21	
## [8]	{1033,1096} => {13426}	0.002563774	0.2083333	0.01230611	11.862835	20	
## [9]	{1096} => {1086}	0.009101397	0.1983240	0.04589155	7.893498	71	
## [10]	{1096} => {1046}	0.008460454	0.1843575	0.04589155	7.732114	66	

Vamos a ver una representación gráfica a modo de grafo de las nuevas reglas de asociación encontradas:

```
## Available control parameters (with default values):
## layout      = stress
## circular    = FALSE
## ggraphdots   = NULL
## edges        = <environment>
## nodes        = <environment>
## nodetext    = <environment>
## colors       = c("#EE0000FF", "#EEEEEEFF")
## engine       = ggplot2
## max         = 100
## verbose     = FALSE
```



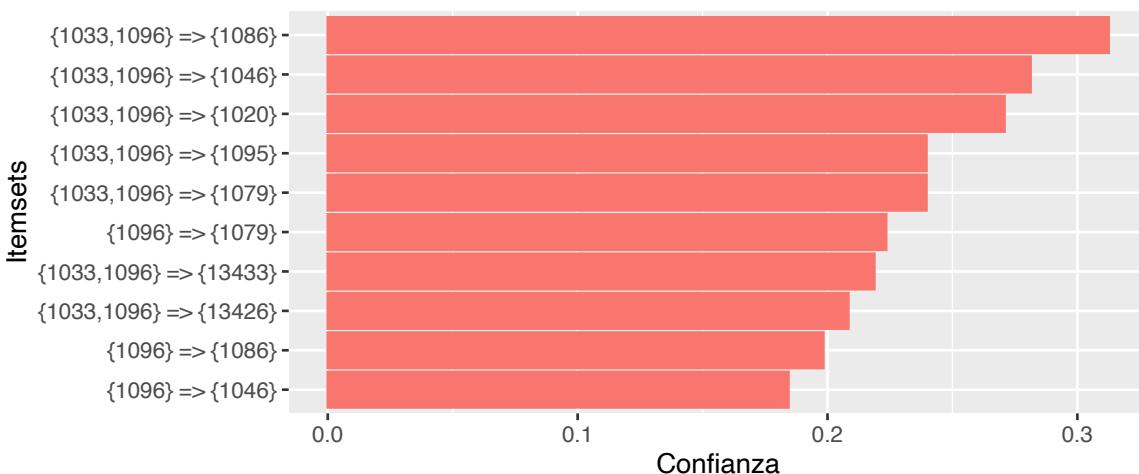
Finalmente, se han obtenido 10 reglas, y como vemos en el grafo, la mayoría de ellas consisten en dos productos en el antecedente, los productos 1033 y 1096, por tanto, como habíamos observado en el estudio de itemsets frecuentes, la venta conjunta de estos dos productos se hace de manera frecuente.

Podemos observar con un valor más alto de soporte, la regla que consiste en la venta de estos dos productos que lleva a la venta del producto 1079. Además, las reglas en las que el consecuente son los productos 1086, 1020 y 1046, tienen unos valores de lift bastante altos, indicando así que se trata de reglas robustas.

Vamos a ordenar las reglas en función de los distintos parámetros:

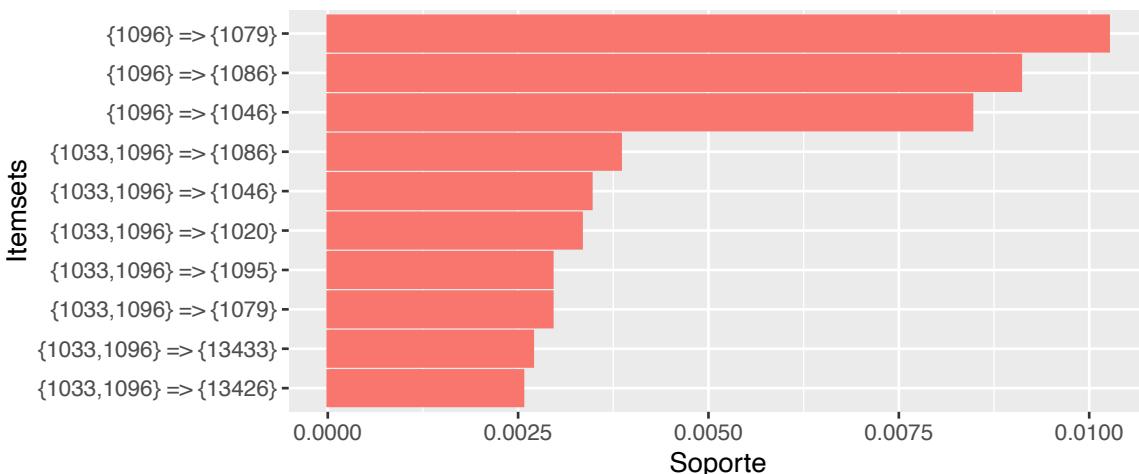
- Ordenación según confianza:

Itemsets según el parámetro de confianza



- Ordenación según soporte (frecuencia con que los objetos son comprados juntos):

Itemsets según el parámetro de soporte



Con esta segunda ordenación observamos lo siguiente: los artículos que se han vendido juntos con mayor frecuencia son el 1096 y el 1079, con una frecuencia del 1.0 % del total de transacciones. También destacar que la compra de los productos 1096 y 1086 en la misma transacción ha tenido lugar en un 0.91 % de las transacciones. Además, la venta de los productos 1096 y 1046 se ha producido con una frecuencia del 0.84 %. Para el resto de itemsets no tienen una frecuencia ni del 0.35 %.

Los valores del soporte son tan bajos debido al gran número de transacciones, por lo tanto, para que una transacción tenga un valor de soporte del 1 % se ha tenido que producir un total de 79 veces.

5.1.4.3. Evaluación de las reglas

Veamos un resumen de las reglas encontradas, con las siguientes métricas:

- Support: 0.0019228
- Confidence: 0.18

```
## set of 10 rules
##
## rule length distribution (lhs + rhs):sizes
```

```
## 2 3
## 3 7
##
##      Min. 1st Qu. Median     Mean 3rd Qu.     Max.
##      2.00    2.25    3.00    2.70    3.00    3.00
##
## summary of quality measures:
##      support      confidence      coverage      lift
##      Min. :0.002564  Min. :0.1844  Min. :0.01231  Min. : 6.538
## 1st Qu.:0.002948  1st Qu.:0.2109  1st Qu.:0.01231  1st Qu.: 7.772
## Median :0.003397  Median :0.2315  Median :0.01231  Median : 9.537
## Mean   :0.004961  Mean   :0.2377  Mean   :0.02238  Mean   : 9.726
## 3rd Qu.:0.007307  3rd Qu.:0.2630  3rd Qu.:0.03750  3rd Qu.:11.846
## Max.   :0.010255  Max.   :0.3125  Max.   :0.04589  Max.   :12.438
##
##      count
##      Min. :20.0
## 1st Qu.:23.0
## Median :26.5
## Mean   :38.7
## 3rd Qu.:57.0
## Max.   :80.0
##
## mining info:
##      data ntransactions      support confidence
## TransBasket          7801 0.00192283        0.18
```

Si los valores de support y confidence están próximos a los ajustados, revisar.

Métricas:

- Soporte:
 - Valor medio: 0.002564
 - Valor mínimo: 0.002179
 - Valor máximo: 0.010255
- Confianza:
 - Valor medio: 0.2377
 - Valor máximo: 0.3125
 - Valor mínimo: 0.1844

El valor de lift mide la importancia y robustez de una regla:

- Lift:
 - Valor medio: 9.726
 - Valor máximo: 12.438
 - Valor mínimo: 6.538

Hemos obtenido unos valores de lift bastante altos, lo que indica que nuestras reglas son improtantes y robustas. Como ya estudiamos en la descripción teórica, este parámetro indica la fuerza de la asociación entre los productos de la parte de la izquierda (antecedentes) y los de la derecha. Cuanto mayor sea el valor de lift, mayor evidencia tendremos de que la regla no se deba a la aleatoriedad, sino que se trata de un patrón de comportamiento existente.

La regla mas robusta que hemos encontrado es la siguiente: al comprar los productos 1033 y 1096, se comprará también el 1086. Esta transacción ha ocurrido un total de 30 veces (frecuencia del 0.38 %). Su valor de lift es de 12.43, que es un valor bastante alto, indicando así que el producto 1086 (consecuente) está bastante vinculado a la compra conjunta de los productos 1033 y 1096 (antecedentes).

```
##      lhs          rhs      support    confidence coverage     lift      count
## [1] {1033,1096} => {1086} 0.003845661 0.3125      0.01230611 12.43782 30
```

La transacción que más veces se ha repetido ha sido:

```
##      lhs          rhs      support    confidence coverage     lift      count
## [1] {1096} => {1079} 0.0102551 0.2234637 0.04589155 7.713452 80
```

lhs	rhs	support	confidence	coverage	lift	count
[1] {1096}	=> {1079}	0.0102551	0.2234637	0.0458916	7.713452	80

Comprar el producto 1079 al comprar el 1096, en un total de 80 ocasiones y con un valor de lift de 7.7, confirmando así la robustez de esta regla.

Otra métrica interesante para cuantificar la calidad de las reglas de asociación obtenidas es el *Test exacto de Fisher*, que permite contrastar las siguientes hipótesis:

$$\begin{cases} H_0 : \text{La regla obtenida se debe al azar, es un resultado aleatorio} \\ H_1 : \text{La regla obtenida se debe a un patrón de comportamiento real} \end{cases}$$

```
## # A tibble: 6 x 7
##   rules          support  confidence coverage     lift count Testfisher
##   <fct>        <dbl>      <dbl>      <dbl> <dbl> <int>      <dbl>
## 1 {1033,1096} => {1086} 0.00385 0.312 0.0123 12.4      30  1.77e-25
## 2 {1033,1096} => {1046} 0.00346 0.281 0.0123 11.8      27  2.98e-22
## 3 {1033,1096} => {1020} 0.00333 0.271 0.0123 12.2      26  8.02e-22
## 4 {1033,1096} => {1095} 0.00295 0.240 0.0123 10.8      23  4.42e-18
## 5 {1033,1096} => {1079} 0.00295 0.240 0.0123 8.27     23  1.84e-15
## 6 {1096} => {1079} 0.0103 0.223 0.0459 7.71     80  1.49e-51
```

Conclusión: En la columna Testfisher se encuentran los *p-valores* del contraste, y son todos menores que $\alpha = 0.05$, por lo que no existen evidencias significativas a favor de que las reglas obtenidas son fruto del azar, por lo que podemos afirmar que significativamente, las reglas de asociación obtenidas se deben a un comportamiento real de ventas.

5.1.4.4. Reglas maximales

Un itemset es *maximal* si no existe otro itemset que sea su superset. Se dice *regla maximal* a aquella que es generada con un itemset maximal. Vamos a estudiar la presencia de este tipo de reglas en las obtenidas haciendo uso de la función *is.maximal()*

```
## set of 7 rules
```

Existen 7 reglas maximales:

```
##      lhs          rhs       support     confidence   coverage    lift      count
## [1] {1033,1096} => {13426} 0.002563774 0.2083333 0.01230611 11.862835 20
## [2] {1033,1096} => {1020}   0.003332906 0.2708333 0.01230611 12.212548 26
## [3] {1033,1096} => {1046}   0.003461095 0.2812500 0.01230611 11.795867 27
## [4] {1033,1096} => {1095}   0.002948340 0.2395833 0.01230611 10.803408 23
## [5] {1033,1096} => {1086}   0.003845661 0.3125000 0.01230611 12.437819 30
## [6] {1033,1096} => {1079}   0.002948340 0.2395833 0.01230611 8.269865 23
## [7] {1033,1096} => {13433}  0.002691963 0.2187500 0.01230611 6.538194 21
## Testfisher
## [1] 1.336665e-16
## [2] 8.017800e-22
## [3] 2.978559e-22
## [4] 4.418948e-18
## [5] 1.767117e-25
## [6] 1.837553e-15
## [7] 3.844979e-12
```

	lhs	rhs
[1]	{1033,1096}	=> {13426}
[2]	{1033,1096}	=> {1020}
[3]	{1033,1096}	=> {1046}
[4]	{1033,1096}	=> {1095}
[5]	{1033,1096}	=> {1086}
[6]	{1033,1096}	=> {1079}
[7]	{1033,1096}	=> {13433}

5.1.4.5. Reglas redundantes

Se dice que dos reglas son *redundantes* si tienen el mismo antecedente y mismo consecuente. Se trata de reglas que son subconjuntos de otras reglas más grandes.

Por ejemplo, dadas dos reglas de asociación:

1. $\{A, B\} \rightarrow \{C\}$
2. $\{A\} \rightarrow \{C\}$

Se tiene que la regla 1 es redundante.

Vamos a estudiar la presencia de este tipo de reglas en las obtenidas haciendo uso de la función *is.redundant()*.

```
## set of 0 rules
```

No existen reglas redundantes.

5.1.5. Conclusiones

Con el estudio y el análisis de las transacciones se ha observado el siguiente patrón de venta: en la mayoría de transacciones se han vendido uno u dos productos diferentes.

También se ha observado que hay ciertos productos que ocupan un volumen de ventas bastante alto, por ejemplo, la venta conjunta de los productos 1033 y 1096 se ha producido en un 1.23 % del conjunto de todas las transacciones. Este valor puede parecer bajo, pero teniendo en cuenta que contamos con 7801 transacciones, supone que la venta de estos artículos se ha producido unas 170 veces, por lo que es bastante frecuente.

El algoritmo nos ha mostrado reglas para aquellos productos que se más se venden. Observando los valores obtenidos del parámetro *lift*, podemos afirmar que estas reglas son todas bastante robustas, es decir, las reglas no se deben a la aleatoriedad o al azar, sino que se trata de un patrón de comportamiento de ventas real.

5.2. Proceso de la ciencia de datos

Este apartado lo dedicaremos a realizar un proceso de ciencia de datos completo, teniendo en cuenta los siguientes objetivos:

- Analizar los datos proporcionados para conocer como varían las ventas de dos productos lácteos con el tiempo.
- Demostrar que existe la posibilidad de construir buenos modelos para predecir el volumen futuro de venta de productos a partir de los datos.
- Desarrollo de modelos para predecir ventas.

5.2.1. Lectura y descripción de los datos

Los datos contienen información correspondiente a ventas de dos productos lácteos (uno con calcio y otro sin calcio) durante un período de 5 meses, desde el 1 de Septiembre de 2020 hasta el 30 de Enero de 2021, obteniéndose un total de 140025 observaciones y se estructuran de la siguiente forma: cada fila corresponde a la línea de un ticket y hace referencia a la venta de un artículo en particular.

En este conjunto de datos inicial encontramos las siguientes variables:

- **Id de ticket:** Variable numérica que identifica únicamente a cada ticket de venta.
- **Línea de ticket:** Variable numérica con la línea correspondiente del ticket.
- **Fecha:** Fecha en que se realizó la venta.
- **Código:** Identificador del producto.
- **Cantidad:** Número de items vendidos de un determinado producto.
- **Precio:** Precio base del artículo libre de impuestos, euros.
- **Precio con impuestos:** Precio de venta del artículo, en euros.
- **Descuento:** Descuento aplicado.
- **Importe:** Importe de la compra libre de impuestos, en euros.
- **Importe con impuestos:** Importe a pagar por el comprador, en euros.

5.2.2. Preparación de los datos (Preprocesado)

En este paso, vamos a llevar a cabo la limpieza de los datos para su posterior estudio, representación y modelado. En este punto del proceso, trataremos de encontrar, corregir o eliminar registros erróneos en los datos.

5.2.2.1. Transformación de los datos

Hay algunas variables que necesitan ser transformadas, en particular, la variable código, la cantidad de artículos vendidos y linea del ticket han sido transformadas para tenerlas en un formato adecuado. Vamos a visualizar la nueva estructura de los datos:

```
dataset %>% str() # Estructura de los datos trás reformato
```

```
## 'data.frame': 140025 obs. of 10 variables:
##   $ ID_TICKET           : num  22549194 22549215 22549242 22549242 22549264 ...
##   $ LINEA_TICKET        : Factor w/ 89 levels "1","2","3","4",...: 1 1 1 2 3 7 1 2
##   $ FECHA               : Date, format: "2020-08-01" "2020-08-01" ...
##   $ CODIGO              : Factor w/ 2 levels "20445","22336": 2 1 2 2 2 2 2 2 2 1
##   $ CANTIDAD             : int  1 6 6 6 1 1 6 1 1 5 ...
##   $ PRECIO              : num  1.35 1.26 1.35 1.35 1.35 1.35 1.35 1.35 1.35 1.26 ...
##   $ PRECIO_CON_IMPUESTOS: num  1.49 1.39 1.49 1.49 1.49 1.49 1.49 1.49 1.49 1.39 ...
##   $ DESCUENTO            : num  0 0 0 0 0 0 0 0 0 0 ...
##   $ IMPORTE              : num  1.35 7.58 8.13 8.13 1.35 1.35 8.13 1.35 1.35 6.32 ...
##   $ IMPORTE_CON_IMPUESTOS: num  1.49 8.34 8.94 8.94 1.49 1.49 8.94 1.49 1.49 6.95 ...
```

5.2.2.2. Duplicados

Se comprueba la existencia de registros duplicados y concluimos que no existen duplicados.

```
dataset[duplicated(dataset)==TRUE,]
```

```
## [1] ID_TICKET           LINEA_TICKET       FECHA
## [4] CODIGO              CANTIDAD          PRECIO
## [7] PRECIO_CON_IMPUESTOS DESCUENTO        IMPORTE
## [10] IMPORTE_CON_IMPUESTOS
## <0 rows> (or 0-length row.names)
```

5.2.2.3. Datos faltantes

Al estar trabajando con fechas, es muy importante comprobar la uniformidad en los datos, para ello buscaremos la existencia de registros faltantes de la siguiente forma:

```
# Construcción de datos con todas las fechas entre la primera fecha
# y la última de los datos que tenemos
FechasCompletas <- seq(min(dataset$FECHA), max(dataset$FECHA), by = "day")

# Creo un DF de fechas
FechasCompletas <- data.frame(FECHA = FechasCompletas )

# Merge al conjunto de fechas completas y al cijo inicial para añadir NA
```

```
# a aquellos valores faltantes
DatosCompletos <- merge(FechasCompletas, dataset, by = "FECHA",
                           all.x = TRUE)

# Valores faltantes en el conjunto de datos completo
Miss_values <- which(is.na(DatosCompletos$ID_TICKET) == TRUE)
```

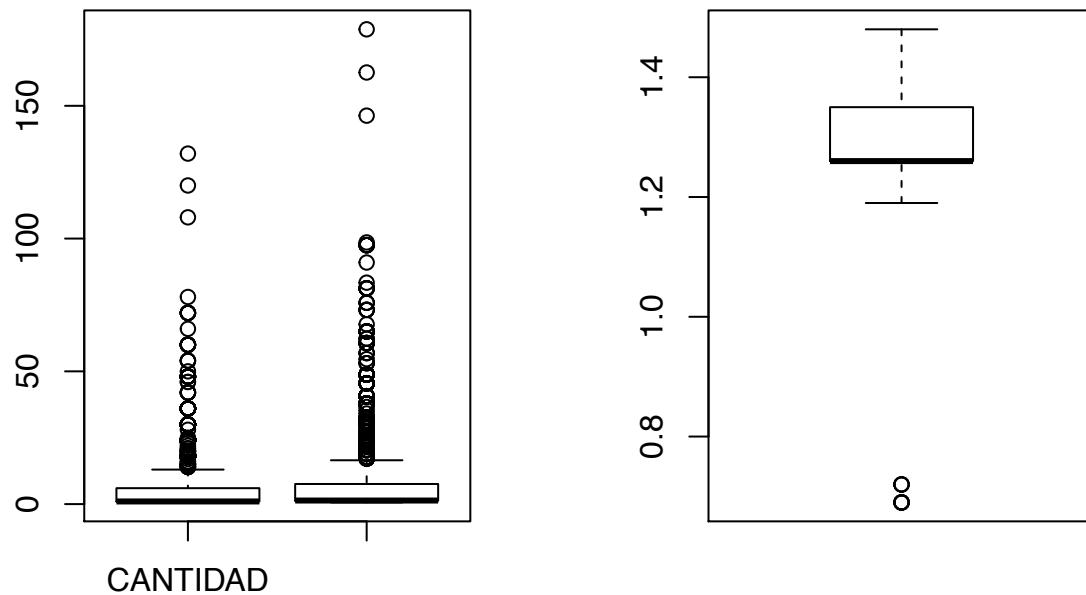
Con registros faltantes nos referimos a que falten las ventas correspondientes a algún día concreto dentro del período de estudio, 1/09/2020-30/01/2022.

Existen un total de 2 valores faltantes, que corresponden a un 0.00143 % del total de datos. Se trata de un porcentaje ínfimo del total. En otras condiciones, procederíamos a imputar estos valores, sin embargo, estos días no estaban contemplados en el conjunto de datos inicial debido a que corresponden a festivos: 2020-12-25, el día de Navidad y 2021-01-01, año nuevo. Por tanto, el motivo de la falta de datos es el cierre del establecimiento y por ello, podemos continuar con nuestro análisis haciendo uso del conjunto de datos inicial.

5.2.2.4. Outliers

Estudiamos valores atípicos para las variables *precio con impuestos*, *cantidad* e *importe con impuestos*, ya que estudiarlos para el resto no tiene mucho interés.

Cantidad e importe (impuestos) **Precio (impuestos)**



Para la variable precio con impuestos, un importe menor de 1.2€ se podría considerar un valor extremo, existiendo dos valores atípicos entre los datos con unos precios de venta de 79 y 76 céntimos. Para la variable cantidad, se considerará un valor atípico la compra de 14 o más artículos. Por último, respecto a la variable importe con impuestos, una compra de más de 18.07€ podemos considerarla como una compra con un valor extremo.

5.2.2.5. Creación de variables

Variables temporales

Se ha considerado oportuno la extracción de la siguiente información como nuevas variables temporales: día de la semana, semana del año, mes y año de cada instancia a partir de la variable *fecha* y haciendo uso de la librería lubridate. De esta forma, se podrá hacer un análisis del comportamiento de ventas teniendo en cuenta distintas granularidades tratando de entender cómo afecta la temporalidad a la venta de productos.

```
dataset$ANO      <- year(dataset$FECHA) # Extracción del año  
dataset$MES      <- month(dataset$FECHA) # Extracción del mes  
dataset$DIA       <- day(dataset$FECHA) # Extracción del día  
dataset$SEMANA_ANO <- week(dataset$FECHA) # Extracción semana del año  
dataset$DIA_SEMANA <- wday(dataset$FECHA,week_start = 1 )
```

Tipo de producto

Además de las variables temporales, vamos a añadir una nueva variable: *TIPO*, que representa el tipo de producto lácteo, siendo esta una variable categórica que toma dos posibles valores: *Con calcio* o *sin calcio*. El producto que no lleva calcio es aquel con identificador 20445

5.2.2.6. Datasets de entrada de modelos

La variable objetivo es el volumen de ventas diario, por lo que necesitamos transformar el conjunto de datos inicial en uno que tenga una variable *ventas* con el número total de items que se venden diariamente. De esta forma, obtendremos tres conjuntos de datos: Uno con el volumen diario total, otro con el volumen de ventas diario del producto sin calcio y un tercer conjunto para el producto con calcio. De esta forma, podremos modelar el volumen de ventas en cada ocasión y posteriormente hacer comparaciones de la suma de las predicciones de los productos por separado con la predicción de la venta de la suma de ambos productos.

Los conjuntos de datos para el entrenamiento de modelos predictivos constan de las siguientes variables: fecha, precio medio con impuestos, descuento medio, año, mes, día, semana del año y día de la semana.

Para los modelos de Regresión Lineal es necesario tener un conjunto de datos con variables *dummy* para el día de la semana y el mes del año, con la intención de representar la pertenencia de cada instancia a los distintos grupos. Posteriormente, es necesario factorizar estas nuevas variables, ya que es algo imprescindible para ejecutar el modelo.

Para la creación de variables *dummy*, definimos una función *crea_col_dummy*, que hace uso de la función *mutate* y *spread*.

```
crea_col_dummy <- function(DataFrame, Columna) {  
  DataFrame %>%  
    mutate(valor = 1) %>%  
    spread(key = Columna, value = valor, fill = 0)  
}  
  
VolVentas_FECHA_poiss =  
  crea_col_dummy(VolVentas_FECHA_poiss,"DIA_SEMANA")  
  
VolVentas_CALCIO_FECHA_poiss=
```

```

crea_col_dummy(VolVentas_CALCIO_FECHA_poiss, "DIA_SEMANA")

VolVentas_SIN_CALCIO_FECHA_poiss=
  crea_col_dummy(VolVentas_SIN_CALCIO_FECHA_poiss,
                 "DIA_SEMANA")

VolVentas_FECHA_poiss =
  crea_col_dummy(VolVentas_FECHA_poiss, "MES")

VolVentas_CALCIO_FECHA_poiss=
  crea_col_dummy(VolVentas_CALCIO_FECHA_poiss, "MES")

VolVentas_SIN_CALCIO_FECHA_poiss=
  crea_col_dummy(VolVentas_SIN_CALCIO_FECHA_poiss, "MES")

```

Una vez tenemos los cuatro conjuntos de datos, tres con los correspondientes volúmenes de ventas diarios y el dataset inicial con el pre procesado necesario, procedemos a guardarlos en formato RData para poder acceder a ellos en cualquier momento que sea necesario.

```

save(dataset, file = "Datos/Dataset_Final.RData")
save(VolVentas_SIN_CALCIO_FECHA, file = "Datos/VENTAS_Dia_SCALCIO.RData")
save(VolVentas_CALCIO_FECHA, file = "Datos/VENTAS_Dia_CALCIO.RData")
save(VolVentas_FECHA, file = "Datos/VENTAS_Dia_TOTAL.RData")

save(VolVentas_FECHA_poiss,
      file = "Datos/VENTAS_Dia_TOTAL_poiss.RData")
save(VolVentas_CALCIO_FECHA_poiss,
      file = "Datos/VENTAS_Dia_CALCIO_poiss.RData")
save(VolVentas_SIN_CALCIO_FECHA_poiss,
      file = "Datos/VENTAS_Dia_SCALCIO_poiss.RData")

```

5.2.3. Análisis exploratorio de datos (EDA)

Una vez hemos realizado el preprocesamiento de los datos necesario, procedemos a la fase del análisis exploratorio.

Este apartado lo dedicaremos a hacer un análisis profundo de las ventas, añadiendo gráficos que muestren el comportamiento del consumidor.

Como tenemos datos correspondientes a ventas, trataremos de responder a las siguientes cuestiones:

- ¿Cuál es el patrón de venta de cada producto? ¿Se venden las mismas unidades, o destaca la venta de uno de ellos?
- ¿Cómo varían las ventas en función del tiempo?
- ¿Qué variables podrían influir más a la hora de vender un producto?

5.2.3.1. Resumen de los datos

```
VolVentas_FECHA[,c(1:6)] %>% summary() # Resumen de los datos

##      FECHA          VENTAS    NUM_TRANSACCIONES PRECIO_MEDIO_IMPUESTOS
##  Min.   :2020-08-01   Min.   : 30   Min.   : 17.0   Min.   :1.418
##  1st Qu.:2020-09-15  1st Qu.:1811  1st Qu.: 472.0  1st Qu.:1.437
##  Median :2020-10-30 Median :2154   Median : 587.0  Median :1.439
##  Mean    :2020-10-30 Mean   :1978   Mean   : 536.7  Mean   :1.439
##  3rd Qu.:2020-12-14 3rd Qu.:2481   3rd Qu.: 680.0  3rd Qu.:1.441
##  Max.   :2021-01-30 Max.   :8011   Max.   :1440.0  Max.   :1.449
##      DESCUENTO_MEDIO    IMPORTE_MEDIO_IMPUESTOS
##  Min.   :0.000000   Min.   :1.716
##  1st Qu.:0.006196  1st Qu.:3.433
##  Median :0.031703  Median :3.601
##  Mean   :0.102185  Mean   :3.580
##  3rd Qu.:0.062889  3rd Qu.:3.752
##  Max.   :4.875000  Max.   :5.139
```

Observamos lo siguiente:

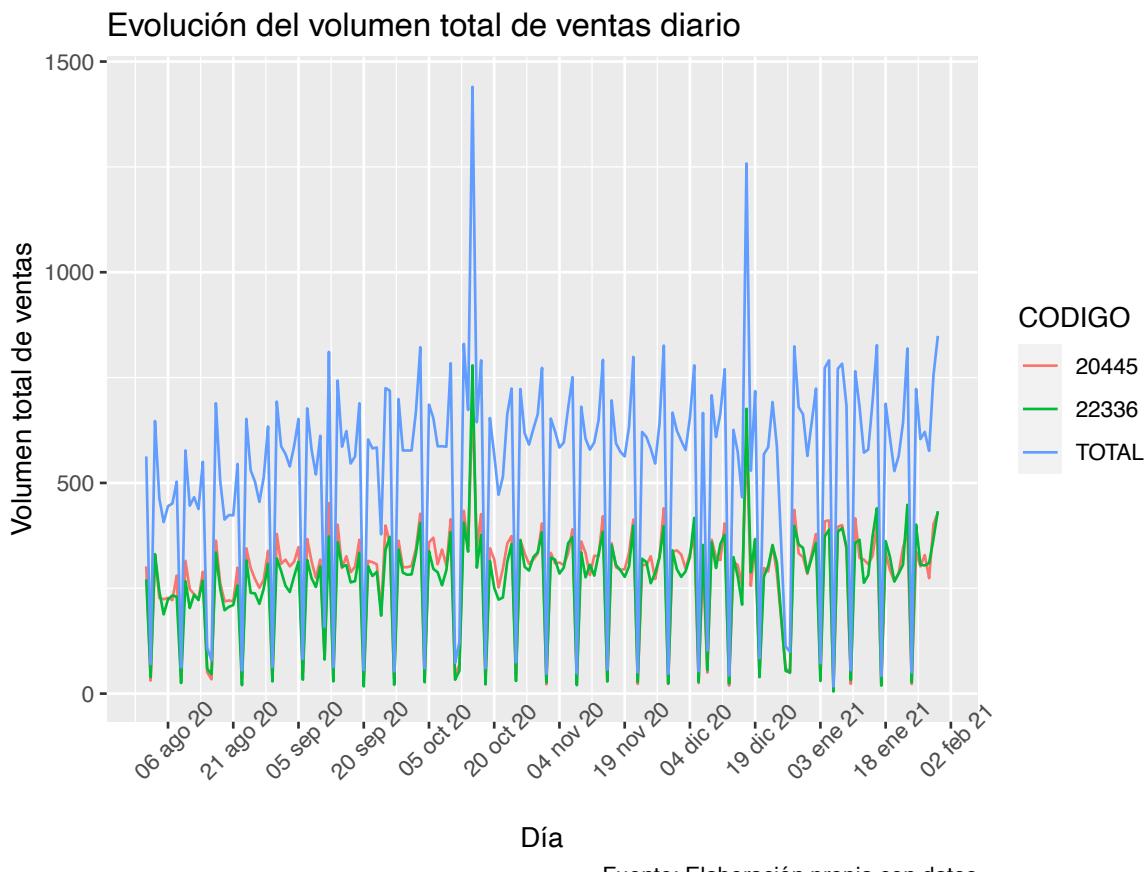
- La cantidad media diaria de productos vendidos es de 1978 unidades, con una media diaria de 537 transacciones
- El precio medio (con impuestos) de los productos es de 1.44€, siendo el importe medio diario de venta de 3.58€
- El descuento medio no llega a los 10 céntimos de euro.

Analizando el número de items vendidos y el importe de venta, es interesante mencionar que de media, se han vendido cuatro items por transacción, con un precio medio de 4.25€. El 75 % de las ventas ha sido de seis items o menos, habiendo transacciones donde el número de productos ascendía hasta las 132 unidades. Con respecto al importe de venta (con impuestos), éste ha variando desde los 69 céntimos a los 196.68€. Sin embargo únicamente el 25 % de las transacciones han sido de más de 8.34€.

Las 140025 filas encontradas en el conjunto de datos corresponden a 97143 ventas diferentes.

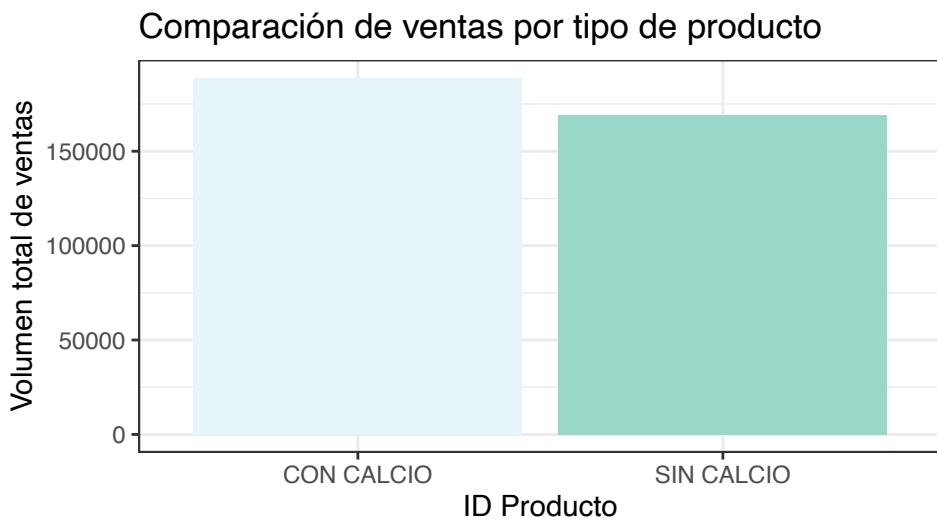
5.2.3.2. Representaciones gráficas

En primer lugar, podemos ver gráficamente la evolución del volumen total de ventas diarias:



En el gráfico podemos apreciar como el volumen de ventas diario fluctúa bastante en función del día, encontrándolas en un rango entre 17 y 1440 ventas diarias. Hay dos momentos donde el volumen de ventas es considerablemente superior al resto, a mediados del mes de Octubre de 2020 y a mediados de Diciembre de este mismo año. Se observa un patrón muy marcado, con picos de muy pocas ventas y otros donde el volumen sube bastante para ambos productos, sin embargo, podemos afirmar que las ventas son ligeramente superiores para el producto con calcio (22336).

A continuación, vamos a hacer una comparación del volumen de ventas total por productos:



El volumen de ventas del producto con calcio ha sido ligeramente superior, con un volumen total de ventas de 188867 unidades frente a las 169196 unidades vendidas del producto que no lleva calcio.

El importe de venta del producto con calcio es de 1.49€ (precio con impuestos) y el producto que no lleva calcio tiene un precio de venta de 1.39€. Esto nos conduce a que los usuarios no han optado siempre por la compra de la opción más económica, sino que han comprado en un mayor número de ocasiones el producto con calcio.

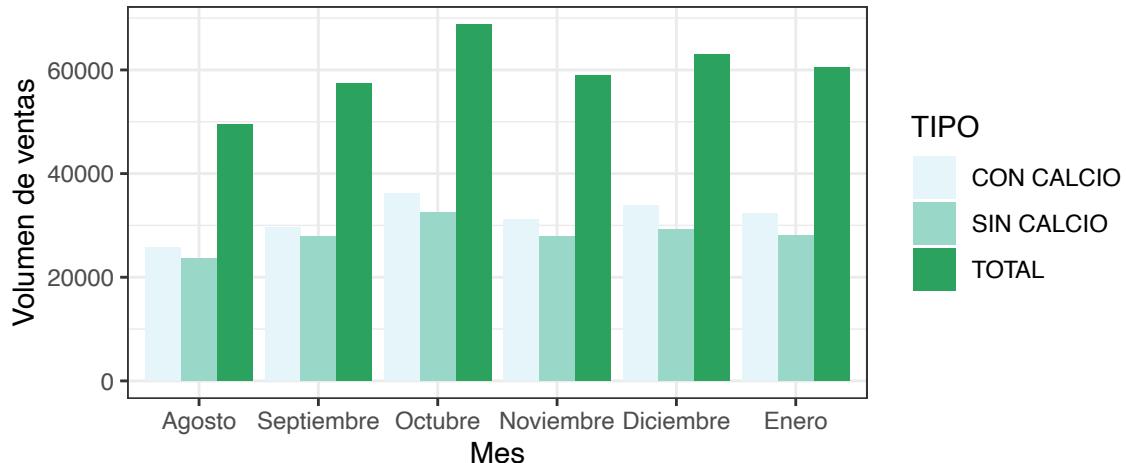
En la tabla que se muestra a continuación, vemos que en las transacciones donde aparece el producto con calcio, han tenido un importe medio de venta ligeramente superior que en las que aparece el producto sin calcio.

Tabla 5.2: Importe medio de venta y precio por producto.

Tipo producto	Importe medio transacción	Precio medio producto
Con calcio	4.07	1.49
Sin Calcio	3.31	1.39

Para tratar de entender mejor el comportamiento de venta, vamos a estudiar la evolución de los valores de ventas en función del día de la semana y mes del año.

Nº total de artículos vendidos según mes del año

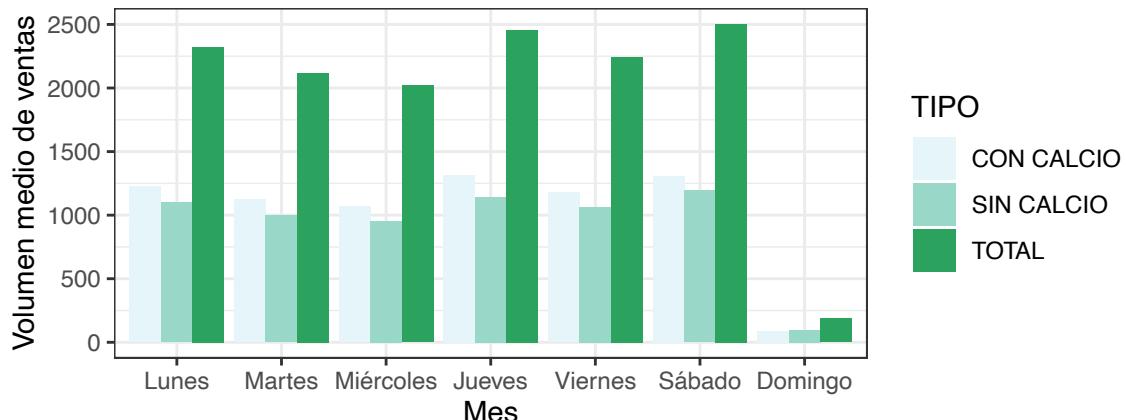


Fuente: Elaboración propia con datos de ventas diarias

Inicialmente, podemos ver una tendencia creciente del número de ventas total, donde se tiende a vender más cada mes. Sin embargo, en el mes de Octubre se alcanza un máximo de artículos vendidos, con un total de 68805 artículos. En los meses siguientes el comportamiento fué de una disminución de la venta, con posterior subida del número de artículos vendidos y luego otro descenso. La tendencia general de venta de artículos es creciente, la clientela tiende a comprar más productos cada vez. Respecto a las ventas de cada producto individual, todos los meses ha sido mayor el volumen de venta del artículo con calcio, alcanzándose en el mes de Octubre el mayor número de artículos vendidos de cada tipo, superando en ambos casos las treinta y dos mil unidades vendidas.

ID_TICKET	LINEA_TICKET	CODIGO	CANTIDAD	PRECIO_CON_IMPUESTOS	DESCUENTO	IMPORTE_CON_IMPUESTOS
22551535	2	22336	6	1.49	5.03	8.49

Comparación del número medio de artículos vendidos Según día de la semana



Fuente: Elaboración propia con datos de ventas diarias

Observando el gráfico, vemos que el Sábado es el día de la semana donde el número de artículos vendidos es mayor, con un total de 2323 items. Por el contrario, el Domingo no se superan ni los 250 artículos vendidos, en media. Cabe destacar que los Domingos se ha vendido, de media, mayor número de items sin calcio, comportamiento que no había ocurrido hasta el momento. Los Jueves es un día donde también se tiende a vender gran cantidad de productos.

5.2.3.2.1. Variable precio

El precio medio de venta del producto que no lleva calcio es de 1.26€, habiendo variado entre los 0.69 y los 1.38 €.

Para el artículo con calcio el precio medio es algo superior: 1.35€, con un precio mínimo de 0.72 € y un máximo de 1.48 euros.

Vamos a estudiar esta variable, tratando de averiguar si esta variación se debe a la época del año o al número de artículos comprado, ya que podría haber promociones para tratar de impulsar las ventas donde los precios pudieran verse afectados.

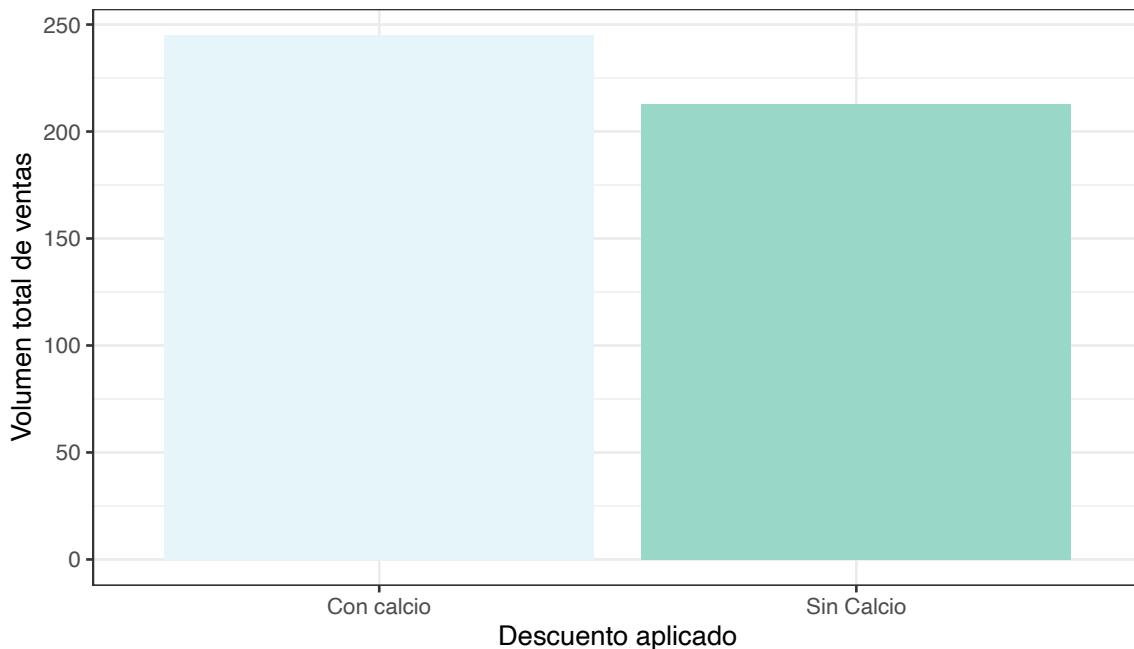
Los precios de los productos varían debido a descuentos aplicados a todos los artículos de una misma compra en función del importe de la misma, por ejemplo, si se superan los 50€, se hará un 10 % de descuento, y por tanto, habremos pagado menos por cada artículo. Sin embargo, la variable descuento, se aplica a cada artículo de manera individual, por ejemplo, en la transacción con identificador 22551535 se vendieron 6 unidades del producto 22336 por un precio de 1.49€ (impuestos incluidos) cada unidad, lo que supone un total de 8.94€ por las 6 unidades, pero se ha aplicado un descuento del 5.03 %.

Se ha aplicado descuento en un total de 1278 artículos, siendo el descuento medio aplicado del 10.26 % respecto del importe total de la venta. Estos descuentos han sido aplicados durante todo el período temporal estudiado. El mayor descuento aplicado ha sido del 50 %, aplicado en dos ocasiones a mediados de Diciembre. En el gráfico mostrado a continuación, se presenta un histograma comparativo del volumen de ventas total de cada producto en función del descuento habiendo dividido los datos en tres intervalos de tal manera que se ha tomado la distribución de los deciles de esta variable:

```
##      0%   10%   20%   30%   40%   50%   60%   70%   80%   90%  100%
## 3.66  5.00  9.95 10.00 10.00 10.00 10.00 10.00 10.00 14.99 50.00
```

Lo que podemos observar es que donde se acentúa más la diferencia del volumen de ventas es al aplicar los mayores descuentos, en un 10 % del total de observaciones.

Descuento del 14.99% o más



5.2.3.3. Grado de asociación de las variables

En la matriz de correlación mostrada a continuación podemos ver el grado de asociación de las diferentes variables entre sí y con la variable objetivo, *ventas*.

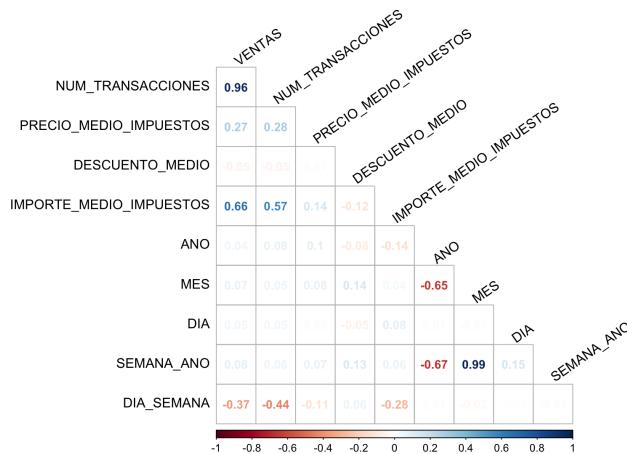


Figura 5.1: Matriz de correlación. Fuente: elaboración propia con datos de ventas.

Observamos como el grado de asociación positiva entre la variable número de transacciones diaria y la variable objetivo, ventas diarias (número de items) es muy elevado, indicando así que conforme más transacciones se hayan producido, mayor número de artículos se venderán. También tienen una correlación importante la variable volumen de

ventas e importe medio con impuestos de la transacción. El comportamiento es similar en los conjuntos de datos individuales para cada producto.

5.2.4. Modelado

A continuación, se expone la aplicación de los modelos predictivos estudiados con el objetivo de predecir la demanda de productos diaria a partir de las siguientes variables explicativas: precio medio con impuestos, descuento medio, año, mes y día de la semana.

Para cada algoritmo, se construirán tres modelos:

- Predicción del volumen de total de ventas diario (producto con calcio y sin calcio)
- Predicción del volumen de ventas diario para el producto con calcio
- Predicción del volumen de ventas diario para el producto sin calcio

Posteriormente, realizaremos una comparación de las predicciones de la suma de productos con la suma de las predicciones proporcionadas por cada modelo individual.

Para los diferentes modelos entrenados, se recogerá su rendimiento para posteriormente compararlos y elegir el mejor modelo para predecir la demanda.

5.2.4.1. Modelos estadísticos clásicos

Partición de los datos en los modelos de regresión:

Se ha tomado una partición de 80 % 20 % para datos de entrenamiento y testeo, con el objetivo de entrenar el modelo para posteriormente estudiar su rendimiento y capacidad de generalización.

De esta manera, para mantener la temporalidad de los datos, tomamos los 145 primeros registros para entrenar el modelo y 36 para el testeo. Esto corresponde a entrenar los modelos con datos diarios desde el 1 de Agosto al 23 de Diciembre, para posteriormente realizar predicciones del 24 de Diciembre al 30 de Enero.

```
n=nrow(VolVentas_FECHA_poiss)
ntest=36
indient = 1:(n-ntest)
inditest= (n-ntest+1):n
```

5.2.4.1.1. Modelo de Regresión de Poisson

Dado que la variable respuesta es discreta y de tipo conteo, se ha elegido este modelo en el que se asume que el volumen de ventas diario sigue una distribución de Poisson.

$$Y \sim Po(\mu), \quad \mu = \text{Número medio de ventas diario}$$

5.2.4.1.1.1. Modelado

En primer lugar, estimaremos los parámetros del modelo de regresión de Poisson, utilizando un conjunto de datos con variables dummy para el día de la semana y el mes del año, con la intención de representar la pertenencia de cada instancia a los distintos grupos.

También entrenaremos un modelo haciendo uso del conjunto de datos con las variables día de la semana y mes en forma de factor, para comprobar que modelo nos da unas mejores métricas.

Las variables explicativas son las siguientes:

- Variables dummy/factorizadas del día de la semana y el mes del año
- Precio medio con impuestos y descuento
- Día de la semana
- Mes del año

Ventas totales

```
ModeloP_TOT_dummy =
  glm(VENTAS~PRECIO_MEDIO_IMPUESTOS*DESCUENTO_MEDIO+
      (LUNES+MARTES+MIERCOLES+JUEVES+VIERNES+SABADO+DOMINGO)+ 
      (AGOSTO+SEPTIEMBRE+OCTUBRE+NOVIEMBRE+DICIEMBRE),
  family = poisson(link = "log"),
  data=Ventas_TOTAL_ENT_poiss)
summary(ModeloP_TOT_dummy)

##
## Call:
## glm(formula = VENTAS ~ PRECIO_MEDIO_IMPUESTOS * DESCUENTO_MEDIO +
##       (LUNES + MARTES + MIERCOLES + JUEVES + VIERNES + SABADO +
##       DOMINGO) + (AGOSTO + SEPTIEMBRE + OCTUBRE + NOVIEMBRE +
##       DICIEMBRE), family = poisson(link = "log"), data = Ventas_TOTAL_ENT_poiss)
##
## Deviance Residuals:
##    Min      1Q   Median      3Q     Max
## -53.003  -5.544    0.460    3.716   73.763
##
## Coefficients: (2 not defined because of singularities)
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 -17.837991  1.158990 -15.391 < 2e-16
## PRECIO_MEDIO_IMPUESTOS        16.124932  0.806001  20.006 < 2e-16
## DESCUENTO_MEDIO                -13.215728  3.719804 -3.553 0.000381
## LUNES1                         2.432609  0.016816 144.658 < 2e-16
## MARTES1                        2.330825  0.016832 138.480 < 2e-16
## MIERCOLES1                     2.354630  0.016836 139.856 < 2e-16
## JUEVES1                        2.550898  0.016707 152.685 < 2e-16
## VIERNES1                       2.379075  0.016888 140.872 < 2e-16
## SABADO1                         2.547224  0.016740 152.161 < 2e-16
## DOMINGO1                         NA          NA          NA          NA
## AGOSTO1                          -0.271368  0.006420 -42.269 < 2e-16
## SEPTIEMBRE1                      -0.098212  0.006260 -15.689 < 2e-16
## OCTUBRE1                          0.034655  0.006068  5.711 1.12e-08
## NOVIEMBRE1                        -0.070297  0.006495 -10.824 < 2e-16
## DICIEMBRE1                         NA          NA          NA          NA
## PRECIO_MEDIO_IMPUESTOS:DESCUENTO_MEDIO  9.199825  2.584555  3.560 0.000372
##
```

```

## (Intercept) ***
## PRECIO_MEDIO_IMPUESTOS ***
## DESCUENTO_MEDIO ***
## LUNES1 ***
## MARTES1 ***
## MIERCOLES1 ***
## JUEVES1 ***
## VIERNES1 ***
## SABADO1 ***
## DOMINGO1 ***
## AGOSTO1 ***
## SEPTIEMBRE1 ***
## OCTUBRE1 ***
## NOVIEMBRE1 ***
## DICIEMBRE1 ***
## PRECIO_MEDIO_IMPUESTOS:DESCUENTO_MEDIO ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
## Null deviance: 93081 on 144 degrees of freedom
## Residual deviance: 25604 on 131 degrees of freedom
## AIC: 26958
##
## Number of Fisher Scoring iterations: 4

```

En la salida, podemos ver en la columna *estimate* la estimación de los coeficientes de regresión para las distintas variables, indicando, para las variables numéricas, el cambio medio en el número de ventas que se produciría si aumentáramos en una unidad esa variable, y para las variables discretas, el cambio medio que provocaría en el número de ventas el que la variable tomara o no ese valor. También encontramos una columna para el error estándar, el valor del estadístico Z y el p-valor. Un p-valor menor que $\alpha = 0.05$ indicará que podemos considerar significativa esa variable de cara a predecir el volumen de ventas. Si por el contrario el p-valor es mayor que α , concluiremos que dicha variable no influye en el volumen de ventas.

Todas las variables introducidas en el modelo influyen significativamente en el volumen de ventas:

- Con respecto al día de la semana, sabemos que el día de la semana que más ventas hay es el Sábado, por lo que estudiamos como afectan las ventas si es o no este día del fin de semana. $\hat{\beta}_{\text{Sábado}} = 2.379075 > 0$, es decir, el volumen de ventas aumentará, en media $e^{2.379075} = 11$ unidades si la compra se realiza un Sábado
- El mes donde hubo más ventas fué durante el mes de Octubre, y según este modelo el volumen de ventas aumentará, en media, manteniendo el resto de variables constante en $e^{0.034655} = 1$ unidad si la compra se hace durante este mes.

Además, la interacción entre las variables descuento y precio medio también es significativa, es decir, la asociación que existe entre ambas varía en función de los diferentes valores que tomen.

Vamos ahora a entrenar el modelo para las variables factorizadas:

```
ModeloP_TOT_factor= glm(VENTAS~PRECIO_MEDIO_IMPUESTOS*DESCUENTO_MEDIO+
                         DIA_SEMANA,
                         family = poisson(link = "log"),
                         data=Ventas_TOTAL_ENT)
#summary(ModeloP_TOT_factor)
```

- La variable *descuento* puede considerarse significativa debido al p-valor = $5.8 \times 10^{-5} < 0.05 = \alpha$ y según la estimación del coeficiente $\hat{\beta}_{descuento} = -14.905524 > 0$ podemos afirmar que un aumento de la variable descuento en una unidad, manteniendo el resto de variables constantes, hará que el volumen de ventas aumente ligeramente.
- Si la venta se produce un Sábado, manteniendo constantes el resto de variables, el volumen de ventas disminuirá en $e^{-2.432920} = 0$ unidad respecto a si se produjera otro día de la semana.

Para seleccionar un modelo adecuado, se ha hecho uso del criterio de información de Akaike, eligiendo aquel que nos da el menor AIC. A continuación se exponen los correspondientes valores del AIC para ambos modelos:

- Modelo con variables dummy: $AIC = 2.6958345 \times 10^4$
- Modelo con variables factorizadas: $AIC = 3.0079583 \times 10^4$

El modelo seleccionado es el modelo para variables dummy, que nos da el menor valor del AIC, indicación de una mayor calidad del modelo estadístico.

A continuación, se procede a entrenar el modelo seleccionado para los datos de testeo haciendo uso de la función *predict*, mostrando las métricas que nos dan el rendimiento del modelo en forma de tabla.

DATOS	R2	RMSE
Datos entrenamiento	0.5745134	689.0828
Datos Test	0.6318605	607.6912

El modelo obtenido explica el 57.45 % de la variabilidad total de los datos en el conjunto de entrenamiento y el 63.19 % para los datos de testeo, no pudiendo considerar este como un buen modelo para explicar el volumen de ventas total. Sin embargo, hay que destacar que el modelo ha sabido generalizar bien con datos nuevos, ya que el coeficiente de correlación ha aumentado, es decir, ha mejorado. Además el RMSE ha disminuido. El valor del error cuadrático medio indica que el modelo se va a equivocar de media, en 608 ventas, que es un valor alto para el volumen de ventas que se está considerando, ya que hay días que se venden menos de 700 items.

Contraste de bondad de ajuste

Para comprobar si se trata o no de un buen modelo, realizamos el contraste de bondad de ajuste, donde se contrastan las siguientes hipótesis:

$$\begin{cases} H_0 : \text{El ajuste lineal es bueno} \\ H_1 : \text{El ajuste no es bueno} \end{cases}$$

El p-valor del contraste: $0 < 0.05 = \alpha$ y por tanto, no existen evidencias significativas para afirmar que el modelo es adecuado.

Ventas de productos con calcio

```
ModeloP_CALCIO_dummy =
  glm(VENTAS~PRECIO_MEDIO_IMPUESTOS+
    (LUNES+MARTES+MIERCOLES+JUEVES+VIERNES+SABADO+DOMINGO)+ 
    (AGOSTO+SEPTIEMBRE+OCTUBRE+NOVIEMBRE+DICIEMBRE) ,
    family = poisson(link = "log"),
    data=Ventas_CALCIO_ENT_poiss)
#summary(ModeloP_CALCIO_dummy)

ModeloP_CALCIO_factor=
  glm(VENTAS~PRECIO_MEDIO_IMPUESTOS+DIA_SEMANA,
    family = poisson(link = "log"),
    data=Ventas_CALCIO_ENT)
#summary(ModeloP_CALCIO_factor)
```

Para ambos modelos podemos afirmar:

- La variable precio medio es altamente significativa para el modelo de datos factorizado, indicando que un aumento de esta variable en un euro indicará una leve disminución de las ventas , manteniendo constantes el resto de constantes. Sin embargo, en el modelo con variables dummy no es significativa.
- El resto de las variables introducidas son significativas al 95 % en el modelo, es decir, influyen en el volumen de ventas.

De nuevo, el modelo seleccionado es el modelo para variables factorizadas, que nos da el menor valor del AIC, indicación de una mayor calidad del modelo estadístico.

- $AIC_{dummies} = 1.6925032 \times 10^4$
- $AIC_{factorizacion} = 1.885407 \times 10^4$

Entrenamos el modelo seleccionado en los datos de entrenamiento haciendo uso de la función *predict* y mostramos en forma de tabla las métricas obtenidas para el conjunto de entrenamiento y el de testeo:

	DATOS_C	R2_C	RMSE_C
Datos entrenamiento	0.5179077	408.0282	
Datos Test	0.5502114	362.1483	

Volvemos a tener unos resultados en los que el modelo ha sabido generalizar bien para datos nuevos, ya que el RMSE ha disminuido en casi 46 unidades y además el valor del coeficiente de determinación ha pasado de explicar el 51.79 de la variabilidad total del volumen de ventas a conseguir explicar un 55.02 % para los datos de testeo.

El p-valor del test de bondad de ajuste: $0 < 0.05 = \alpha$. No existen evidencias significativas para afirmar que se trate de un modelo adecuado.

Ventas de productos sin calcio

```

ModeloP_SIN_CALCIO_dummy =
  glm(VENTAS~PRECIO_MEDIO_IMPUESTOS+
      (LUNES+MARTES+MIERCOLES+JUEVES+VIERNES+SABADO+DOMINGO)++
      (AGOSTO+SEPTIEMBRE+OCTUBRE+NOVIEMBRE+DICIEMBRE) ,
      family = poisson(link = "log"),
      data=Ventas_SIN_CALCIO_ENT_poiss)
#summary(ModeloP_SIN_CALCIO_dummy)

ModeloP_SIN_CALCIO_factor=
  glm(VENTAS~PRECIO_MEDIO_IMPUESTOS+DESCUENTO_MEDIO+
      DIA_SEMANA,
      family = poisson(link = "log"),
      data=Ventas_SIN_CALCIO_ENT)
#summary(ModeloP_SIN_CALCIO_factor)

```

Para ambos modelos podemos afirmar:

- De nuevo, ocurre como para el modelo de productos con calcio. La variable precio medio no va a ser significativa para el volumen de ventas en el modelo con variables dummies pero sí lo es en el modelo con variables factorizadas.
- El resto de las variables son significativas en el modelo a la hora de predecir el volumen de ventas.

De nuevo, el modelo seleccionado es el modelo para variables factorizadas, que nos da el menor valor del AIC, indicación de una mayor calidad del modelo estadístico.

- $AIC_{dummies} = 1.2690964 \times 10^4$
- $AIC_{factorizacion} = 1.3961937 \times 10^4$

Entrenamos el modelo seleccionado en los datos de testeo haciendo uso de la función *predict* y mostramos los resultados a modo de tabla:

DATOS_SC	R2_SC	RMSE_SC
Datos entrenamiento	0.5954031	303.1903
Datos Test	0.5716492	305.4431

En este caso, el modelo no generaliza tan bien como el resto, ya que las métricas obtenidas son peores. Este modelo no es capaz de explicar más que el 57.16 % de la variabilidad del volumen de ventas en los datos de testeo, a pesar de explicar el 59.54 % para los datos de entrenamiento. Los errores cuadráticos medios son muy elevados en comparación con el volumen de ventas que se está prediciendo.

El p-valor del contraste de bondad de ajuste: $0 < 0.05 = \alpha$. No existen evidencias significativas para afirmar que se trate de un modelo adecuado.

Sobredispersión

Estos modelos se han desarrollando asumiendo que la distribución de las ventas diarias sigue una Poisson, caracterizándose esta distribución porque su esperanza y su varianza coinciden; pero esto no siempre ocurre trabajando con conjuntos de datos reales. Se dice

entonces que el modelo presenta sobredispersión. Vamos a contrastar la presencia de sobredispersión en los modelos entrenados haciendo uso de la función *dispersiontest* de la librería *AER*-

```
## 
## Overdispersion test
## 
## data: ModeloP_TOT_factor
## z = 2.4655, p-value = 0.00684
## alternative hypothesis: true dispersion is greater than 1
## sample estimates:
## dispersion
## 211.7527

## 
## Overdispersion test
## 
## data: ModeloP_CALCIO_dummy
## z = 2.2397, p-value = 0.01256
## alternative hypothesis: true dispersion is greater than 1
## sample estimates:
## dispersion
## 121.0532

## 
## Overdispersion test
## 
## data: ModeloP_SIN_CALCIO_dummy
## z = 2.6439, p-value = 0.004098
## alternative hypothesis: true dispersion is greater than 1
## sample estimates:
## dispersion
## 79.9638
```

Concluimos que todos los casos existe dispersión y junto con las conclusiones de los test de bondad de ajuste, podemos afirmar que en este caso, el modelo de regresión de poisson no es adecuado para modelar el volumen de ventas diario.

5.2.4.1.2. Modelo de Regresión Binomial Negativa

Ante el problema de la sobredispersión, trataremos de modelar las ventas diarias según un modelo de regresión binomial negativa. No podemos hacer uso de la función *glm* del paquete base de R debido a que no tiene implementada la opción de esta distribución. Por ello, utilizaremos la función *glm.nb* de la librería *MASS*, que incluye la estimación del parámetro de dispersión θ . Los conjuntos de datos son los mismos que los utilizados en los modelos de regresión de poisson, al igual que las variables explicativas:

- Variables dummy/factorizadas del día de la semana y el mes del año
- Precio medio con impuestos y descuento
- Día de la semana
- Mes del año

5.2.4.1.2.1. Modelado

Ventas totales

```
ModeloBN_TOT_dummy =
  glm.nb(VENTAS~PRECIO_MEDIO_IMPUESTOS+DESCUENTO_MEDIO+
         (LUNES+MARTES+MIERCOLES+JUEVES+VIERNES+SABADO+DOMINGO)+
         (AGOSTO+SEPTIEMBRE+OCTUBRE+NOVIEMBRE+DICIEMBRE),
         data=Ventas_TOTAL_ENT_poiss)
summary(ModeloBN_TOT_dummy)

##
## Call:
## glm.nb(formula = VENTAS ~ PRECIO_MEDIO_IMPUESTOS + DESCUENTO_MEDIO +
##        (LUNES + MARTES + MIERCOLES + JUEVES + VIERNES + SABADO +
##        DOMINGO) + (AGOSTO + SEPTIEMBRE + OCTUBRE + NOVIEMBRE +
##        DICIEMBRE), data = Ventas_TOTAL_ENT_poiss, init.theta = 10.43020791,
##        link = log)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -4.7468  -0.3699   0.0081   0.2886   4.1767
##
## Coefficients: (2 not defined because of singularities)
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) 2.899344  10.169850  0.285  0.7756
## PRECIO_MEDIO_IMPUESTOS 1.677074   7.078366  0.237  0.8127
## DESCUENTO_MEDIO -0.005361   0.060797 -0.088  0.9297
## LUNES1        2.483722   0.101214 24.539 <2e-16 ***
## MARTES1      2.364453   0.100532 23.519 <2e-16 ***
## MIERCOLES1   2.382654   0.099699 23.899 <2e-16 ***
## JUEVES1      2.562844   0.101426 25.268 <2e-16 ***
## VIERNES1     2.421807   0.102892 23.537 <2e-16 ***
## SABADO1       2.580693   0.101538 25.416 <2e-16 ***
## DOMINGO1      NA          NA          NA          NA
## AGOSTO1       -0.211735   0.086597 -2.445  0.0145 *
## SEPTIEMBRE1  -0.062193   0.088838 -0.700  0.4839
## OCTUBRE1      0.038859   0.087305  0.445  0.6562
## NOVIEMBRE1   -0.066480   0.089824 -0.740  0.4592
## DICIEMBRE1    NA          NA          NA          NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(10.4302) family taken to be 1)
##
## Null deviance: 820.00 on 144 degrees of freedom
## Residual deviance: 147.28 on 132 degrees of freedom
## AIC: 2224.9
##
## Number of Fisher Scoring iterations: 1
```

```
##  
##  
##           Theta: 10.43  
##           Std. Err.: 1.22  
##  
## 2 x log-likelihood: -2196.893
```

Con la función *glm.nb*, obtenemos una salida parecida a la del modelo de regresión de poisson salvo por el parámetro de dispersión θ , estimado mediante el método de la máxima verosimilitud, para el cual se obtiene un valor que no es el que aparece en la salida, sino su inversa: $\theta = 0.0958754$.

También podemos ver la estimación de los coeficientes del modelo y el estadístico de desviación, que sigue una distribución chi-cuadrado de 132 grados de libertad y tiene un valor de 147.28. Haciendo uso de este estadístico podemos evaluar la sobredispersión de los datos de la siguiente forma:

$$\frac{D}{gl} \implies \frac{147.28}{132} = 1.116 > 1$$

,

La relación anterior nos indica sobredispersión en los datos.

En este modelo, para un nivel de significación del 95 %, los coeficientes estimados para las variables descuento y precio medio pueden suponerse nulos, siendo en ambos casos el p-valor correspondiente mayor que $\alpha = 0.05$. Ocurre igual para los meses de Septiembre, Octubre y Noviembre, es decir, que el modelo se basa únicamente en el día de la semana para predecir el volumen de ventas.

Vamos ahora a entrenar el modelo para las variables factorizadas:

```
ModeloBN_TOT_factor= glm.nb(VENTAS~PRECIO_MEDIO_IMPUESTOS+  
                           DIA_SEMANA+DESCUENTO_MEDIO,  
                           data=Ventas_TOTAL_ENT)  
summary(ModeloBN_TOT_factor)
```

```
##  
## Call:  
## glm.nb(formula = VENTAS ~ PRECIO_MEDIO_IMPUESTOS + DIA_SEMANA +  
##         DESCUENTO_MEDIO, data = Ventas_TOTAL_ENT, init.theta = 9.707696922,  
##         link = log)  
##  
## Deviance Residuals:  
##      Min        1Q     Median        3Q       Max  
## -4.4764   -0.4112    0.0167   0.3313    4.3840  
##  
## Coefficients:  
##                               Estimate Std. Error z value Pr(>|z|)  
## (Intercept)             4.838368  10.145973  0.477  0.633  
## PRECIO_MEDIO_IMPUESTOS  2.000155   7.050334  0.284  0.777  
## DIA_SEMANA2            -0.098212  0.099321 -0.989  0.323  
## DIA_SEMANA3            -0.076961  0.099488 -0.774  0.439
```

```
## DIA_SEMANA4          0.120020  0.100566  1.193   0.233
## DIA_SEMANA5          -0.036213 0.100514 -0.360   0.719
## DIA_SEMANA6          0.121768  0.099268  1.227   0.220
## DIA_SEMANA7          -2.477266 0.104562 -23.692 <2e-16 ***
## DESCUENTO_MEDIO      -0.006974  0.059160 -0.118   0.906
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(9.7077) family taken to be 1)
##
## Null deviance: 763.94  on 144  degrees of freedom
## Residual deviance: 147.30  on 136  degrees of freedom
## AIC: 2227.4
##
## Number of Fisher Scoring iterations: 1
##
##
## Theta:  9.71
## Std. Err.: 1.13
##
## 2 x log-likelihood: -2207.362
```

La única variable significativa de este modelo es que el haber realizado la compra un Domingo. La estimación del coeficiente es: $\hat{\beta}_{\text{Domingo}} = -2.477266 < 0$ y podemos afirmar lo siguiente: ir a comprar un Domingo, hará que el volumen de ventas disminuya ligeramente.

Para seleccionar un modelo adecuado, se ha hecho uso del criterio de información de Akaike, eligiendo aquel que nos de el menor AIC. A continuación se exponen los correspondientes valores del AIC para ambos modelos:

- Modelo con variables dummy: AIC= 2227.3617597
- Modelo con variables factorizadas: AIC= 2224.8931408

Ambos modelos tienen el mismo valor del AIC, pero seleccionamos el de las variables dummy, ya que hay mayor número de variables significativas en el modelo.

El siguiente paso es entrenar el modelo seleccionado en los datos de testeo haciendo uso de la función *predict*. De nuevo, mostraremos el rendimiento de las métricas a modo de tabla:

Datos	R2	RMSE
Datos entrenamiento	0.5550257	704.8192
Datos Test	0.5862130	625.7098

El modelo obtenido explica el 55.5 % de la variabilidad total de los datos en el conjunto de entrenamiento, y el 58.62 % para los datos de testeo. El modelo ha sabido generalizar para datos nuevos. El valor del error cuadrático medio indica que el modelo se va a equivocar de media, en 626 ventas, que es un valor alto para el volumen de ventas que se está considerando, ya que hay días que se venden menos de 700 items.

Ventas de productos con calcio

```

ModeloBN_CALCIO_dummy =
  glm.nb(VENTAS~PRECIO_MEDIO_IMPUESTOS+
         (LUNES+MARTES+MIERCOLES+JUEVES+VIERNES+SABADO+DOMINGO)+
         (AGOSTO+SEPTIEMBRE+OCTUBRE+NOVIEMBRE+DICIEMBRE) ,
         data=Ventas_CALCIO_ENT_poiss)
#summary(ModeloBN_CALCIO_dummy)

ModeloBN_CALCIO_factor=
  glm(VENTAS~PRECIO_MEDIO_IMPUESTOS+DIA_SEMANA,
      data=Ventas_CALCIO_ENT)
#summary(ModeloBN_CALCIO_factor)

```

- Para ambos modelos podemos afirmar que el mes del año no influye en el volumen de ventas
- En el modelo con variables factorizadas volvemos a comprobar que la única variable significativa al 95 %, y por tanto, la única cuya coeficiente no es nulo es si la compra se ha realizado o no un Domingo.
- Sobredispersión del modelo con variables dummies: $\frac{D}{gl} \Rightarrow \frac{147.51}{133} = 1.109098 > 1$.
Existe sobredispersión.

De nuevo, el modelo seleccionado es el modelo que utiliza variable dummies, que nos da el menor valor del AIC, indicación de una mayor calidad del modelo estadístico.

- $AIC_{dummies} = 2044.2612537$
- $AIC_{factorizacion} = 2187.5941504$

Entrenamos el modelo seleccionado en los datos de entrenamiento haciendo uso de la función *predict*

En el siguiente gráfico podemos ver una representación de los valores observados respecto de los valores ajustados.

Datos	R2	RMSE
Datos entrenamiento	0.5137814	409.8795
Datos Test	0.5638640	348.7042

El modelo vuelve a mejorar sus métricas para los datos de testeo. Este modelo es capaz de explicar el 0.563864 % de la variabilidad del volumen de ventas para los datos de testeo, a pesar de explicar el 0.5137814 % de la variabilidad para los datos de entrenamiento. A pesar de obtener estos resultados, el error cuadrático medio cometido es menor en los datos de testeo, pero aún bastante alto para el volumen de ventas diario de los productos con calcio.

Ventas de productos sin calcio

```

ModeloBN_SIN_CALCIO_dummy =
  glm.nb(VENTAS~PRECIO_MEDIO_IMPUESTOS+

```

```
(LUNES+MARTES+MIERCOLES+JUEVES+VIERNES+SABADO+DOMINGO)+  

(AGOSTO+SEPTIEMBRE+OCTUBRE+NOVIEMBRE+DICIEMBRE),  

  data=Ventas_SIN_CALCIO_ENT_poiss)  

# summary(ModeloBN_SIN_CALCIO_dummy)

ModeloBN_SIN_CALCIO_factor=  

  glm.nb(VENTAS~PRECIO_MEDIO_IMPUESTOS+DESCUENTO_MEDIO+  

         DIA_SEMANA,  

         data=Ventas_SIN_CALCIO_ENT)  

# summary(ModeloBN_SIN_CALCIO_factor)
```

Se han obtenido resultados muy similares al resto:

- Para ambos modelos podemos afirmar que el precio medio del artículo no influye de manera significativa en el volumen de ventas.
- En el modelo con variables dummies, únicamente influye el día de la semana y que sea o no el mes de agosto.
- En este caso, para el modelo con variables factorizadas existen dos variables significativas que sea Domingo.
- Sobredispersión del modelo: $\frac{D}{gl} \Rightarrow \frac{147.79}{133} = 1.111203 > 1$. Existe sobredispersión.
Nota: estos grados de libertad son los del modelo con variables dummies, pero en el segundo modelo, el estadístico de desviación tiene 133 grados de libertad, y por tanto, el resultado es el mismo.

El modelo seleccionado es el modelo que utiliza variable dummies, que nos da el menor valor del AIC, indicación de una mayor calidad del modelo estadístico.

- $AIC_{dummies} = 2014.7917081$
- $AIC_{factorizacion} = 2017.900612$

Entrenamos el modelo seleccionado en los datos de entrenamiento haciendo uso de la función *predict*.

Datos	R2	RMSE
Datos entrenamiento	0.5881986	305.9139
Datos Test	0.5845958	293.5128

Este modelo no es capaz de explicar más que el 58.4595839 % de la variabilidad del volumen de ventas en los datos de testeо, explicando el 58.8198623 % de la variabilidad de ventas para los datos de entrenamiento. Los errores cuadráticos medios son muy elevados en comparación con el volumen de ventas que se está prediciendo, pero esta métrica mejora con las predicciones en los datos de testeо, además, el valor del coeficiente R^2 es muy parecido.

Comparación de las métricas obtenidas

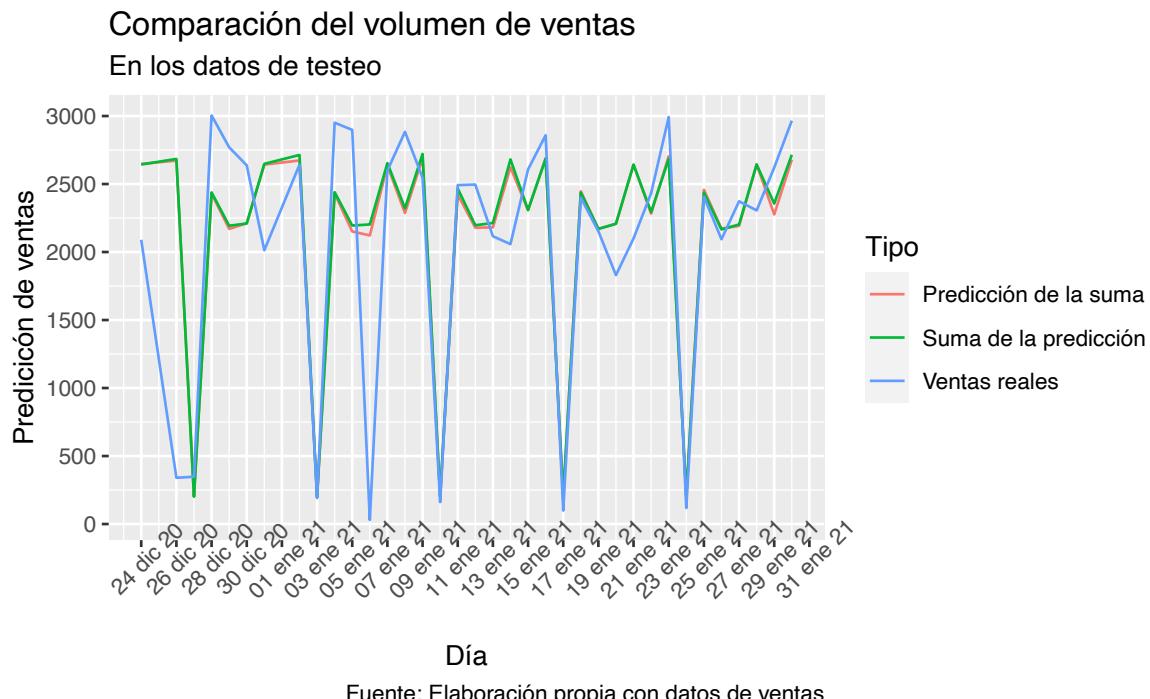
A continuación, mostramos una comparación del rendimiento de los diferentes modelos:

Datos	R2	RMSE
Suma de productos		
Datos entrenamiento	0.5550257	704.8192
Datos Test	0.5862130	625.7098
Producto con calcio		
Datos entrenamiento	0.5137814	409.8795
Datos Test	0.5638640	348.7042
Producto sin calcio		
Datos entrenamiento	0.5881986	305.9139
Datos Test	0.5845958	293.5128

Todos los modelos tienen un rendimiento similar para los datos de testeo y una capacidad de generalización similar, ya que siempre han mejorado las métricas al testear los modelos.

Comparación de las predicciones

Mostramos un gráfico para comparar la predicción de ventas de la suma de productos con la suma de las predicciones de cada uno de los productos por separado:



Se observa que la suma de las predicciones individuales de cada producto arroja unas ventas ligeramente superiores que en la predicción considerando las ventas de ambos productos, pero son prácticamente las mismas predicciones, por lo que no variaría mucho utilizar los modelos para cada producto y sumar las predicciones que usar el modelo que predice la suma del volumen de ventas.

Sin embargo, a pesar de tener este comportamiento similar, vemos que los modelos no han sabido reflejar correctamente las verdaderas variaciones en las ventas cuando la tendencia es creciente. Podemos ver este comportamiento, por ejemplo, los días 31 de

Diciembre, 8, 14 o 20 de Enero. Cabe destacar que esto ocurre cada 8 días, cuando la serie de datos tiene el mismo comportamiento, viene de un fuerte descenso de ventas el Domingo y tiende a subir bruscamente el Lunes, para luego volver a descender el volumen el Miércoles y Jueves, cosa que en las predicciones se ve reflejado como un aumento este cuarto día de la semana.

5.2.4.1.3. Conclusiones

- El modelo de regresión de Poisson no es adecuado para modelar el volumen de ventas diarias, debido a la gran sobredispersión existente en los datos.
- El mejor modelo de regresión Binomial Negativa encontrado es para el volumen de ventas total, donde se consigue explicar un 58.62 % de la variabilidad total de los datos.
- Los RMSE obtenidos son altos en comparación con el volumen de ventas que estamos tratando de predecir.
- El formato de los datos que nos ha proporcionado mejores modelos de regresión binomial es el de las variables dummies
- Emplear un modelo de binomial negativa nos ha llevado a obtener mejores resultados, además de ser mas fiables y precisos que haciendo uso de la distribución de Poisson
- Un posible motivo para no obtener buenos resultados es la limitación que existe en los datos, ya que únicamente tenemos datos de ventas para 181 días, y los modelos no consiguen “aprender” como es el comportamiento de venta en función de las diferentes variables temporales o el precio de venta de los productos.

5.2.4.1.4. Análisis de Series Temporales

Se consideró aplicar un análisis de series temporales debido a la estructura de los datos, ya que este tipo de análisis contempla la estructura temporal de los mismos. Como ya se avanzó en el desarrollo teórico, aplicaremos la metodología Box-Jenkis, la cual tiene en cuenta la dependencia existente de los datos, construyendo así un modelo ARIMA.

Trataremos de modelizar el volumen de ventas total según día de la semana. Para construir la serie, primero hemos añadido los días 25 de Diciembre y 1 de Enero con un número de ventas 0, ya que, si no se tomaba esta decisión, la serie ya no estaría definida según la realidad.

5.2.4.1.4.1. Creación ST y representación de los datos

Los datos que se han utilizado para el análisis han sido las ventas totales para cada día, agrupando los datos según día de la semana, obteniendo así datos con período S=7.

Sin embargo, si construimos la serie con los valores actuales, no podremos aplicarle transformaciones, en particular la transformación de Box-Cox, ya que existen dos valores nulos, las ventas para los días 25 de Diciembre y 1 de Enero. Por este motivo, sumamos una constante a todas las observaciones de modo que sean todas positivas.

Nota: la constante que hemos sumado es de 10 unidades.

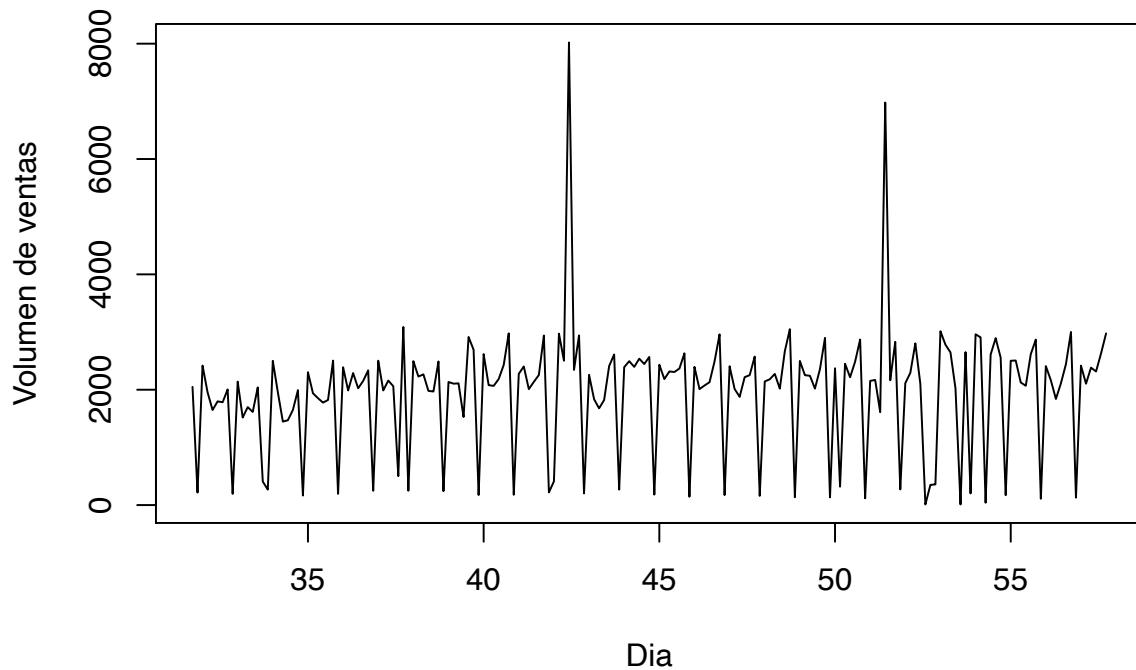
El primer dato, indica que el sábado de la semana 31 del año se vendieron un total de 2049 artículos, aunque en la realidad es que se vendieron 10 unidades menos, pero se le ha sumado una constante a la serie.

```
tsDiaSemanal = ts(Ventas_Totales_Dia_Semana_Completo$ArtVendidos,
                   frequency=7, # Período
                   start=c(31,6) # Semana 36, sábado
)
print(tsDiaSemanal,calendar=TRUE)

##      p1    p2    p3    p4    p5    p6    p7
## 31          2049  216
## 32 2418 1955 1648 1798 1783 2005  194
## 33 2140 1518 1698 1615 2040  408  268
## 34 2501 1957 1447 1470 1649 1993  165
## 35 2304 1938 1853 1776 1822 2505  194
## 36 2390 1988 2289 2025 2153 2335  247
## 37 2503 1987 2159 2060  502 3088  249
## 38 2495 2228 2266 1979 1969 2490  241
## 39 2134 2105 2110 1527 2914 2693  174
## 40 2618 2079 2066 2186 2429 2978  178
## 41 2275 2403 2010 2136 2255 2940  220
## 42  410 2972 2502 8021 2343 2942  202
## 43 2258 1835 1677 1821 2410 2611  268
## 44 2391 2495 2396 2538 2445 2568  181
## 45 2433 2185 2317 2307 2367 2631  144
## 46 2396 2010 2073 2130 2491 2960  173
## 47 2407 2013 1876 2218 2252 2575  159
## 48 2143 2181 2274 2018 2667 3050  136
## 49 2499 2252 2242 2019 2358 2899  133
## 50 2372   319 2450 2216 2486 2870  117
## 51 2153 2169 1611 6979 2164 2830  271
## 52 2113 2292 2804 2100   10  350  358
## 53 3014 2780 2647 2023   10 2652  201
## 54 2961 2908   40 2609 2894 2556  171
## 55 2502 2506 2126 2068 2617 2869  109
## 56 2409 2163 1840 2110 2443 3002  128
## 57 2417 2104 2383 2316 2631 2976
```

Después de haber definido los datos como una serie temporal, visualizamos la evolución de la serie en el tiempo.

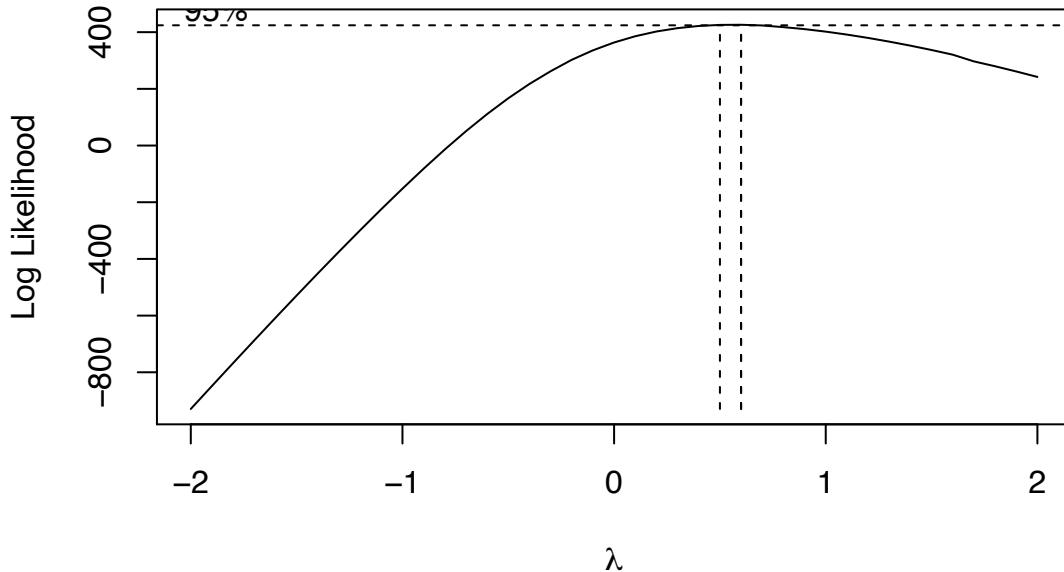
Volumen total de venta por según día de la semana



En el gráfico se puede apreciar cierta estacionalidad de los datos, es decir, movimientos que se repiten regularmente año tras año en los mismo períodos. También observamos que las oscilaciones van aumentando con el tiempo, indicando que la varianza no es constante. Por este motivo, debemos hacer alguna transformación para que la varianza sea constante en el tiempo.

5.2.4.1.4.2. Transformación de BoxCox para estabilizar la varianza

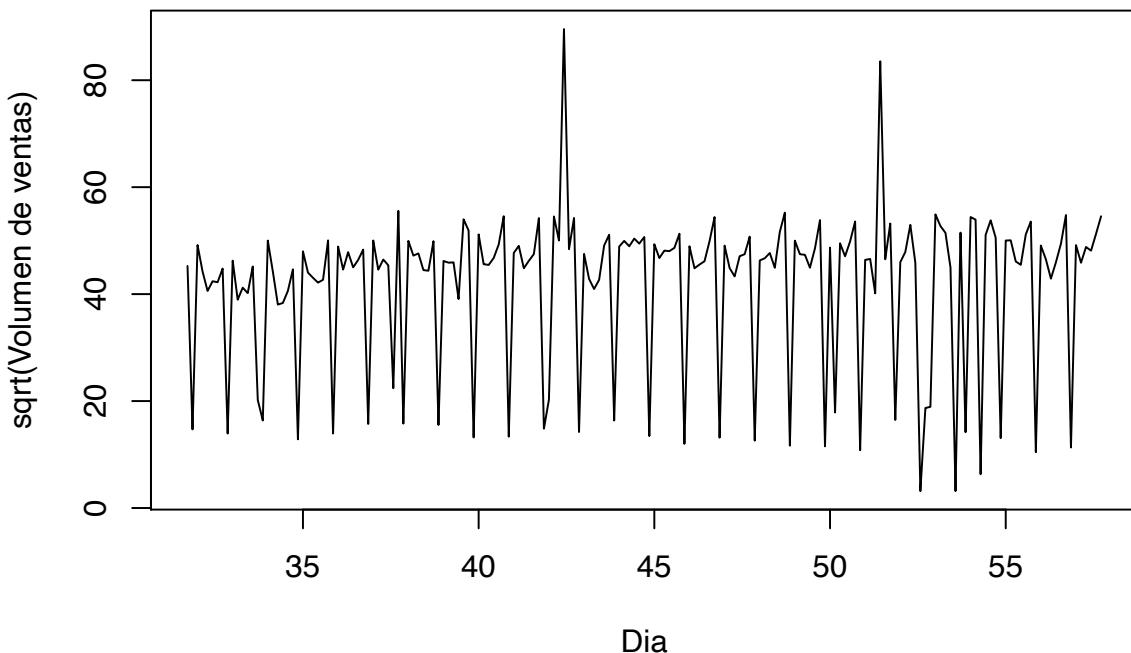
Para encontrar una transformación que haga que la varianza sea constante en el tiempo, haremos uso de la familia de transformaciones Box-Cox con ayuda de la librería *TSA*.



La función *BoxCox.ar* sugiere un óptimo de $\lambda = 0.6$, con un intervalo de confianza al 95 %: (0.5,0.6). Se necesita una transformación sencilla y comprensible, por lo que se ha obtenido por tomar como valor de lambda el extremo inferior del intervalo, $\lambda = 1/2$.

Transformamos los datos y volvemos a representar la serie.

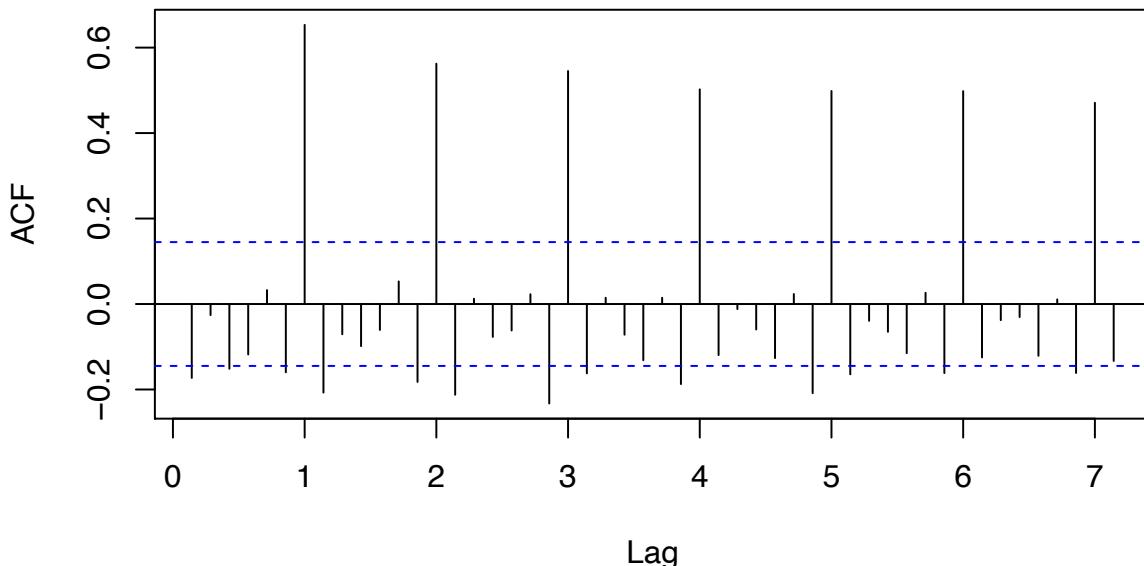
sqrt(Volumen total de venta diario)



5.2.4.1.4.3. Transformaciones para estabilizar la media

Vamos a estudiar si el motivo de la no estacionalidad de los datos en media se debe a que se trata de un proceso integrado. Para ello, hacemos uso de la función de autocorrelación simple.

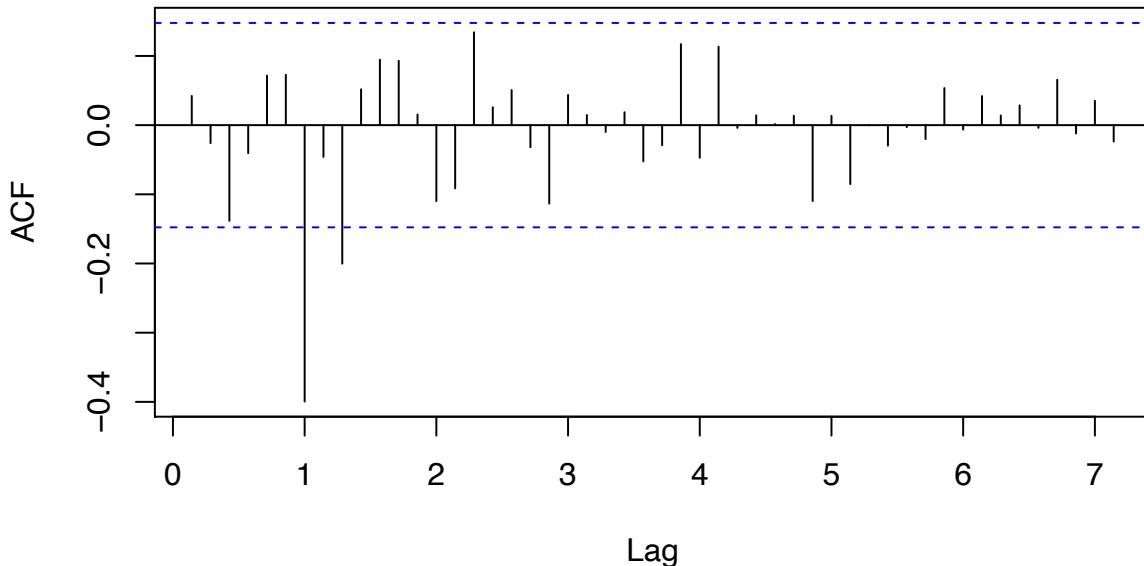
FAS de SQRT de Ventas diarias



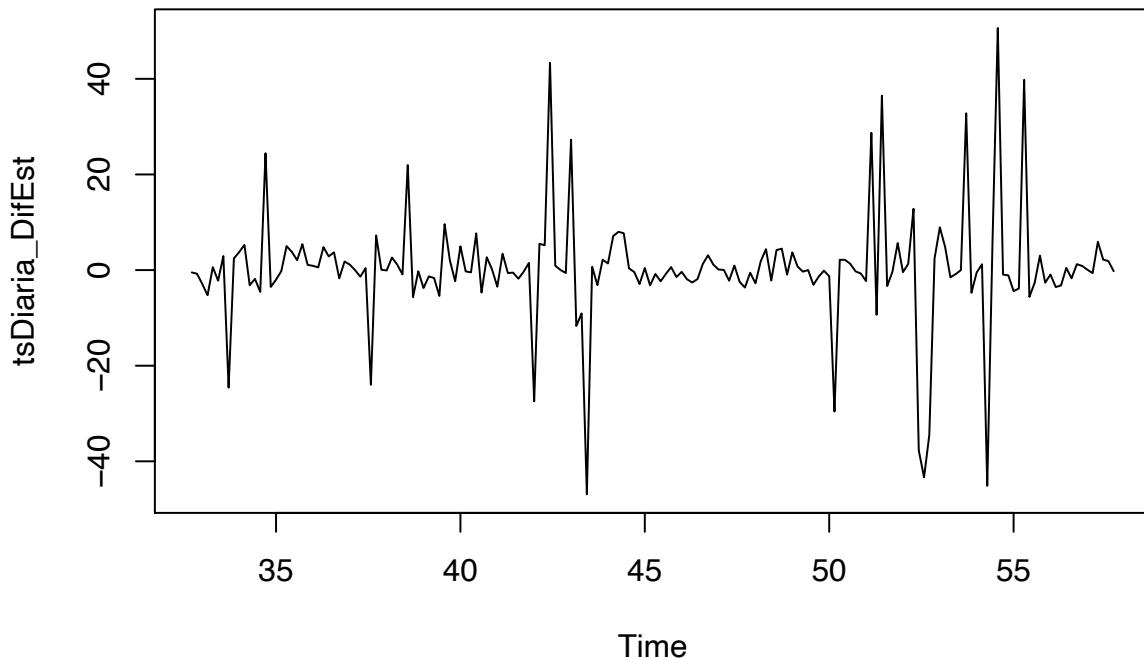
La FAS muestral decrece de lentamente en los retardos estacionales de período 7, indicando que estamos ante un modelo integrado. Debido a esta situación, hacemos una diferencia estacional de la serie y volvemos a representar la FAS ($s=7$).

```
tsDiaria_DifEst = diff(tsDiaSemanal_transf,lag=7,diff=1)
acf(tsDiaria_DifEst, main="FAS de primera diferencia estacional", lag=50)
```

FAS de primera diferencia estacional



Ahora la función de autocorrelación muestral corresponde a la de un proceso estacionario. Por último, representamos gráficamente la serie diferenciada:



Observamos que la serie no muestra ningún comportamiento en particular, sino que se aprecia aleatoriedad, por lo que se podría pensar, que nos encontramos ante un proceso estacionario. Ahora estamos en condiciones de buscar un modelo estacionario para la serie.

5.2.4.1.4.4. Contraste de estacionariedad

Para confirmar la estacionariedad de los datos sugerida con la observación de la gráfica, necesitamos aplicar un test de hipótesis. Aplicamos el test de raíz unitaria de Dikey-Fuller,

donde se contrasta la estacionariedad de los datos a través del siguiente test de hipótesis:

$$\begin{cases} H_0 : \text{ El polinomio autoregresivo tiene una raíz unitaria} \\ H_1 : \text{ Todas las raíces del polinomio autoregresivo son estacionarias} \end{cases}$$

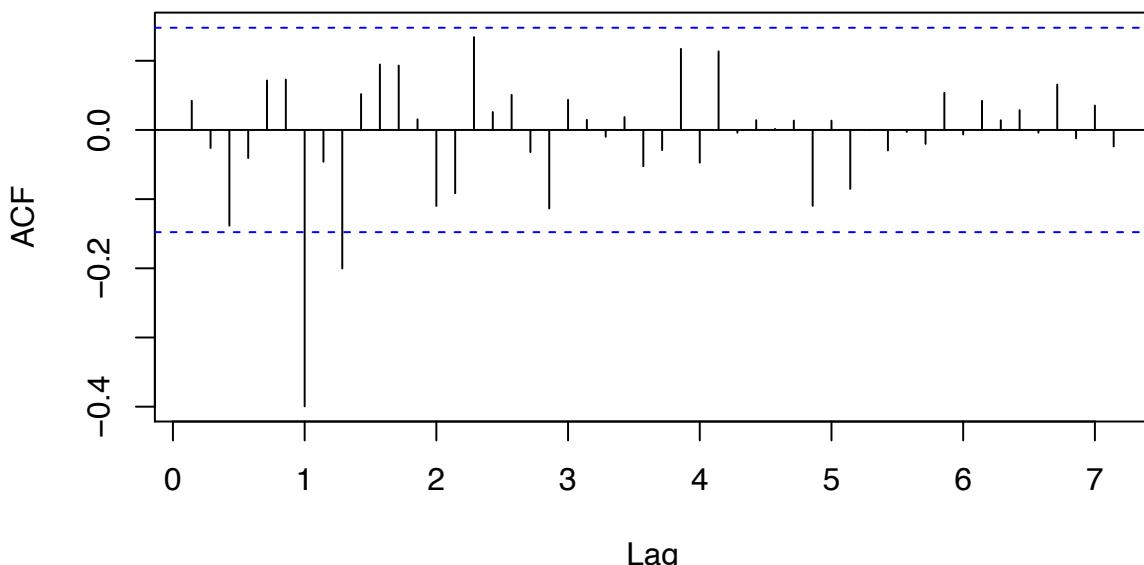
```
## 
## Augmented Dickey-Fuller Test
##
## data: tsDiaria_DifEst
## Dickey-Fuller = -5.1008, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

El p-valor del test= $0.01 < 0.05 = \alpha$, y por tanto concluimos que no existen evidencias significativas para asumir que el polinomio autoregresivo tiene alguna raíz unitaria, la serie es estacionaria.

5.2.4.1.4.5. Identificación de la estructura ARIMA de la serie

Trataremos de identificar la estructura ARIMA más adecuada para esta serie a través de la función de autocorrelación simple (FAC) y de la función de autocorrelación parcial (FAP). Determinar el modelo más adecuado consistirá en e identificar el orden de los procesos de medias móviles y autoregresivos de la componente estacional y la componente regular.

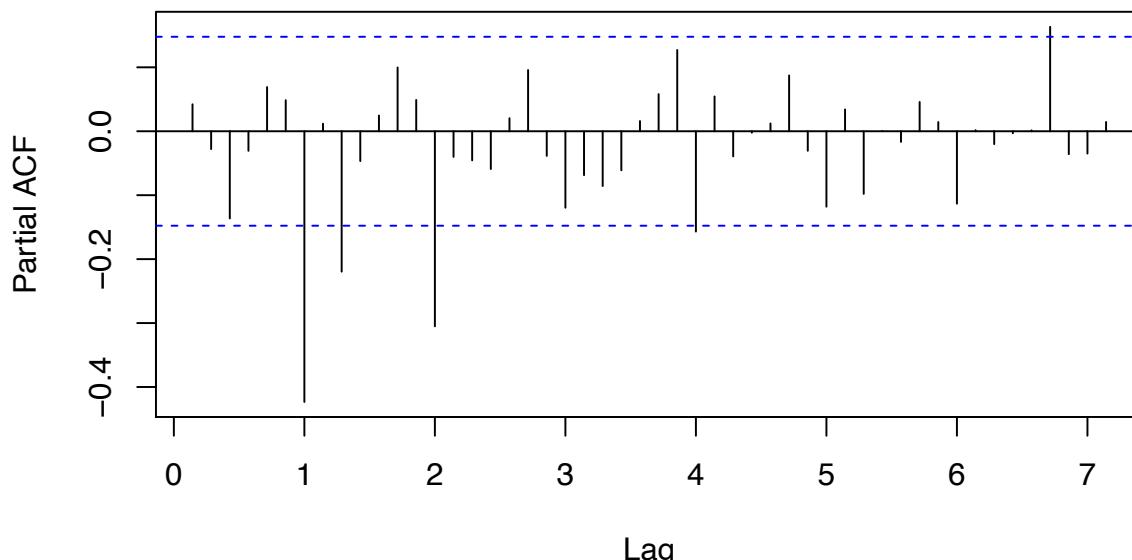
FAS tras una diferencia estacional



- Parte regular: En los primeros retardos no observamos ninguna autocorrelación significativamente no nula, indicando que el modelo tiene una estructura ARMA(0,0) en la parte regular.
- Parte estacional: Observamos una autocorrelación en el primer retardo estacional, por lo que parecería que la parte estacional tiene una estructura $MA(1)_{12}$.

Vamos a comprobar estos supuestos con la FAP.

FAP tras una diferencia estacional



- Parte regular: De nuevo, no hay autocorrelaciones significativamente no nulas en los primeros retardos.
- Parte estacional: En los retardos estacionales, observamos como las autocorrelaciones decrecen rápidamente y a su izquierda, no hay autocorrelaciones significativamente no nulas, lo que avalaría aún más la suposición de un MA(1) en la parte estacional. Modelo propuesto: $MA(1)_{12}$

También observamos como hay otras autocorrelaciones significativamente no nulas, pero esto es debido a que se trata de un intervalo de confianza al 95 %, por lo que cabe esperar que haya algunas autocorrelaciones fuera de las bandas.

El modelo a considerar es un modelo estacional multiplicativo integrado de medias móviles puro: $ARIMA(0, 1, 1)_{12}$

5.2.4.1.4.6. Estimación de parámetros y diagnóstico del modelo

Una vez hemos obtenido un modelo, se han estimado sus parámetros con la función *arima*.

```
Ajuste1 = arima( tsDiaria_DifEst ,# Serie trás una diferencia estacional
                  seasonal = list(order=c(0,0,1),period=7 ))
Ajuste1

##
## Call:
## arima(x = tsDiaria_DifEst, seasonal = list(order = c(0, 0, 1), period = 7))
##
## Coefficients:
##             sma1  intercept
##             -1.0000    0.0898
## s.e.      0.0827    0.0871
##
## sigma^2 estimated as 79.04:  log likelihood = -645.71,  aic = 1295.42
```

Tras comprobar si los coeficientes estimados son o no significativamente nulos, procedemos a eliminar la media del modelo, obteniendo así uno donde todos los coeficientes son significativamente no nulos.

```

##          2.5 %    97.5 %
## sma1     -1.16213365 -0.8378657
## intercept -0.08097887  0.2605421

Ajuste1_1 = arima( tsDiaria_DifEst ,
                    order = c(0,0,0),
                    seasonal = list(order=c(0,0,1),period=7),
                    include.mean = FALSE # Eliminamos la media
)
Ajuste1_1

##
## Call:
## arima(x = tsDiaria_DifEst, order = c(0, 0, 0), seasonal = list(order = c(0,
##   0, 1), period = 7), include.mean = FALSE)
##
## Coefficients:
##           sma1
##           -0.9999
## s.e.    0.1009
## 
## sigma^2 estimated as 79.52:  log likelihood = -646.24,  aic = 1294.47
confint(Ajuste1_1)

##          2.5 %    97.5 %
## sma1     -1.197648 -0.8021465

```

El modelo ajustado corresponde a la siguiente ecuación:

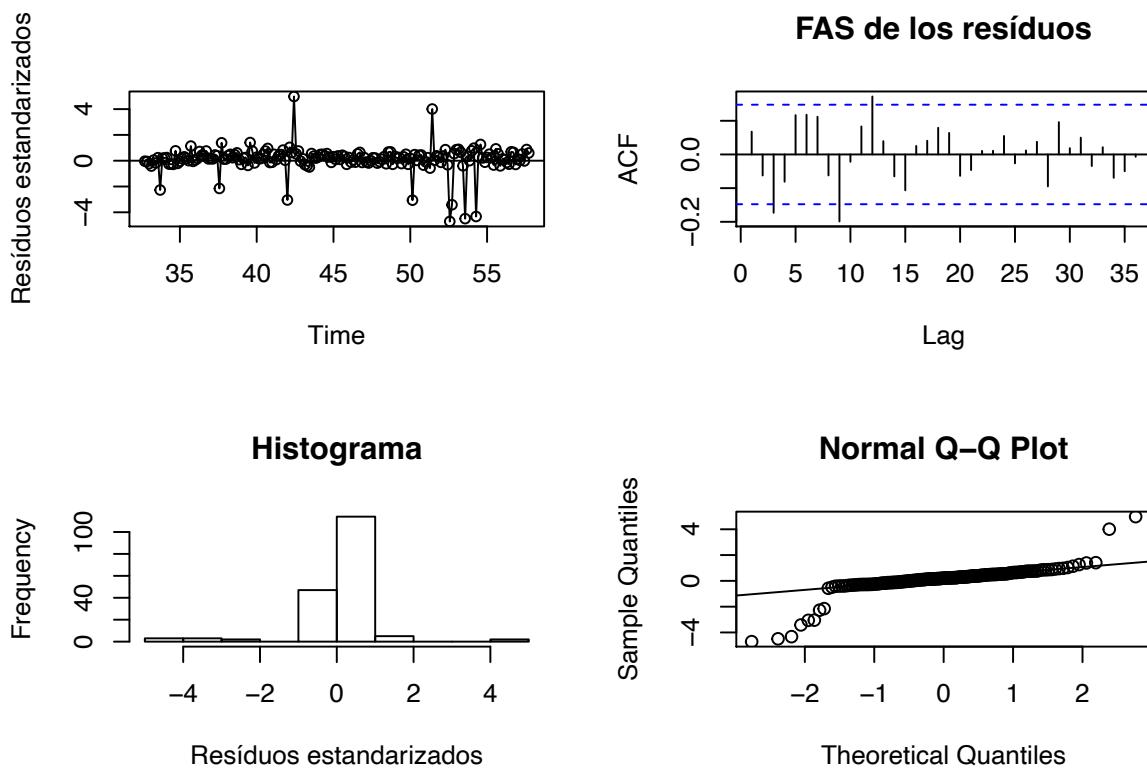
$$Y_t = (1 - L^7)(1 + 0.9999\Theta^7)\alpha_t, \quad \alpha_t \sim RB(0, \sigma^2)$$

Para comprobar si el modelo es o no adecuado, comprobamos su validez a través de la diagnosis de los resíduos y concluimos que este ajuste no es adecuado, ya que según el Test de Ljung-Box, no existen evidencias significativas para aceptar la incorrelación de los resíduos: $p\text{-valor} = 0.002524 < 0.05 = \alpha$. Además, gráficamente podemos observar que los resíduos no se comportan como un ruido blanco.

```

##
## Ljung-Box test
##
## data: Residuals from ARIMA(0,0,0)(0,0,1)[7] with zero mean
## Q* = 31.855, df = 13, p-value = 0.002524
##
## Model df: 1.  Total lags used: 14

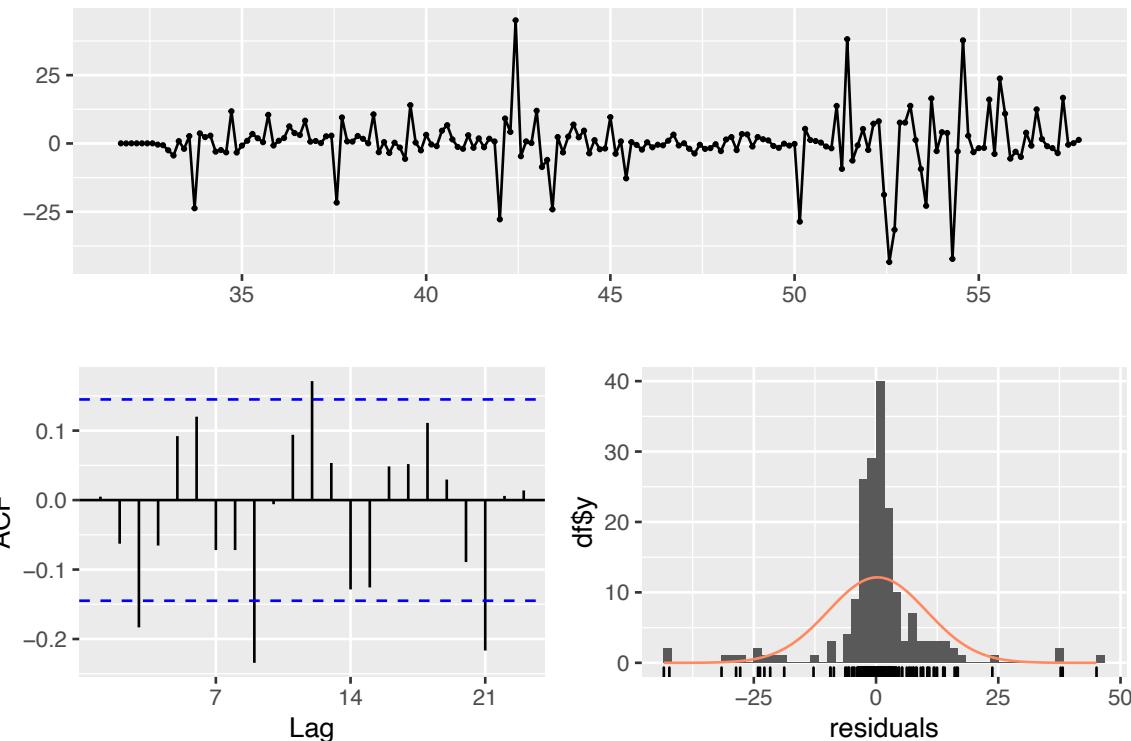
```



Vamos a probar otro modelo, en particular, a través del paquete *forecast* haciendo uso de la función *auto.arima*, que busca un modelo que minimiza el AIC.

```
## Series: tsDiaSemanal_transf
## ARIMA(1,0,0)(2,1,0)[7]
##
## Coefficients:
##             ar1      sar1      sar2
##             0.0709   -0.5180   -0.3068
## s.e.     0.0754    0.0708    0.0694
##
## sigma^2 = 106.2: log likelihood = -660.08
## AIC=1328.17   AICc=1328.4   BIC=1340.85
```

Residuals from ARIMA(1,0,0)(2,1,0)[7]



```
##  
## Ljung-Box test  
##  
## data: Residuals from ARIMA(1,0,0)(2,1,0)[7]  
## Q* = 36.336, df = 11, p-value = 0.0001488  
##  
## Model df: 3. Total lags used: 14
```

El ajuste propuesto es un modelo: $ARIMA(1,0,0)xARIMA(2,0,0)_7$, pero tampoco es adecuado, ya que volvemos a rechazar la hipótesis de incorrelación de los resíduos del Test de Ljung-Box.

5.2.4.1.4.7. Conclusiones

No hemos podido encontrar un modelo adecuado que se ajuste a los datos y que pase la diagnosis, ya que los resíduos no provenían en ningún caso de un proceso de ruido blanco, es decir, no estaban incorrelados entre sí. Por este motivo, al no ser los retardos independientes, un retardo puede guardar cierta relación con otro retardo k períodos después. En estos casos, la autocorrelación puede conducir a una inexactitud en el modelo predictivo, que nos llevaría a interpretaciones erróneas.

La tabla mostrada a continuación expone los diferentes modelos ajutado, el valor del AIC y el p-valor obtenido del test de Ljung-Box. De haber pasado algún modelo la diagnosis, el seleccionado para realizar predicciones del volumen de ventas habría sido aquel con menor valor del AIC.

	MODELO	AIC	p-valor
Modelo 1	ARIMA(0,1,1)_7	1294.474	0.0025242
Modelo 2	ARIMA(1,0,0)x(2,1,0)_7	1328.403	0.0001488
Modelo 3	ARIMA(1,1,1)_7	1293.397	0.0068141
Modelo 4	ARIMA(1,1,0)_7	1341.251	0.0000002

Note:

El p-valor corresponde al test de Ljung-Box

5.2.4.2. Técnicas de aprendizaje automático

El objetivo del presente apartado es predecir el volumen de ventas haciendo uso de los siguientes algoritmos de regresión: máquinas de vector soporte (SVM), KNN y árboles de decisión, en particular XGBoost. Todos estos algoritmos serán aplicados haciendo uso de la función *train* de la librería *caret*. El principal objetivo de usar la misma función para todos los modelos es poder comparar el rendimiento de éstos y seleccionar el mejor algoritmo para predecir el volumen de ventas.

Para el desarrollo de los modelos, haremos uso de los conjuntos de datos de 181 filas generados en el apartado de pre-procesado y limpieza de los datos, siendo las variables predictoras las siguientes mostradas a continuación:

- Precio medio con impuestos
- Descuento medio aplicado
- Mes del año
- Día de la semana

Nota: La variable año, no se ha introducido como predictora del volumen de ventas, ya que su varianza es muy próxima a cero y por tanto, no podríamos entrenar los modelos de aprendizaje automático.

Nota: Para la replicación de los resultados, haremos uso de la función *trainControl*, donde emplearemos validación cruzada con 5 grupos y tres repeticiones. No utilizamos 10 grupos en la validación como es lo usual debido al pequeño número de registros que tenemos.

```
# Para todos los modelos
fitControl <- trainControl(method = "repeatedcv",
                            number = 5, repeats = 3,
                            verboseIter =FALSE )
```

De cara a utilizar optimizar el rendimiento de los algoritmos de regresión, haremos uso de la instrucción *preProcess* de la función *train* para escalar nuestras variables y que estén todas en la misma escala, con el objetivo de obtener mejores métricas y que los modelo minimicen el error al predecir las ventas.

Al hacer la división de los datos para entrenar los modelos, se ha optado por no utilizar un conjunto de datos de validación ya que únicamente tenemos 181 registros y tener datos para validar los modelos supondría tener aún menos registros para entrenarlos, y por tanto, se obtendrían métricas menos precisas.

5.2.4.2.1. División de los datos en entrenamiento y testeo

Al igual que en el modelado clásico, se ha tomado una partición de 80 % 20 % para datos de entrenamiento y testeo, con el objetivo de entrenar el modelo para posteriormente estudiar su rendimiento y capacidad de generalización.

De esta manera, para mantener la temporalidad de los datos, tomamos los 145 primeros registros para entrenar el modelo y 36 para el testeo. Esto corresponde a entrenar los modelos con datos diarios desde el 1 de Agosto al 23 de Diciembre, para posteriormente realizar predicciones del 24 de Diciembre al 30 de Enero.

5.2.4.2.2. Predicción del volumen total de ventas

División de los datos en entrenamiento y testeo

Se toma una partición de 80 % 20 % para los datos de entrenamiento y testeo:

```
# Datos de entrenamiento
DatosEntreamiento_Total <- VolumenVentas_TOTAL[indient,]

# Datos de testeo
DatosTesteo_Total <- VolumenVentas_TOTAL[inditest,]
```

5.2.4.2.2.1. Algoritmo 1: Máquina de vector soporte (SVM)

Hiperparámetros del algoritmo

- Validación cruzada con 5 grupos y tres repeticiones
- Parámetro de costo, C: malla de valores entre 1 y 3. Este parámetro penaliza al modelo por cometer errores. Cuanto mayor sea su valor, menos probable es que el algoritmo realice una predicción errónea.

Modelado

```
# Malla para hiperparámetros
SVMGrid <- expand.grid(C = seq(1,3, length = 20))

set.seed(17)
modeloSVM_T <- train(VENTAS~.,
                      data = DatosEntreamiento_Total[,-c(1,5,6)],
                      method = "svmLinear",
                      trControl=fitControl,
                      preProcess=c("center","scale"),
                      tuneGrid = SVMGrid)
```

Resultados

El modelo tiene un costo $C = 1.3157895$ y nos ofrece las siguientes métricas:

Tabla 5.3: Métricas del mejor modelo SVM. Algoritmo 1.1

	C	RMSE	Rsquared	MAE	RMSESD	RsquaredSD	MAESD
4	1.315789	660.7109	0.6006251	333.5372	431.7381	0.29241	144.3136

Métricas del remuestreo:

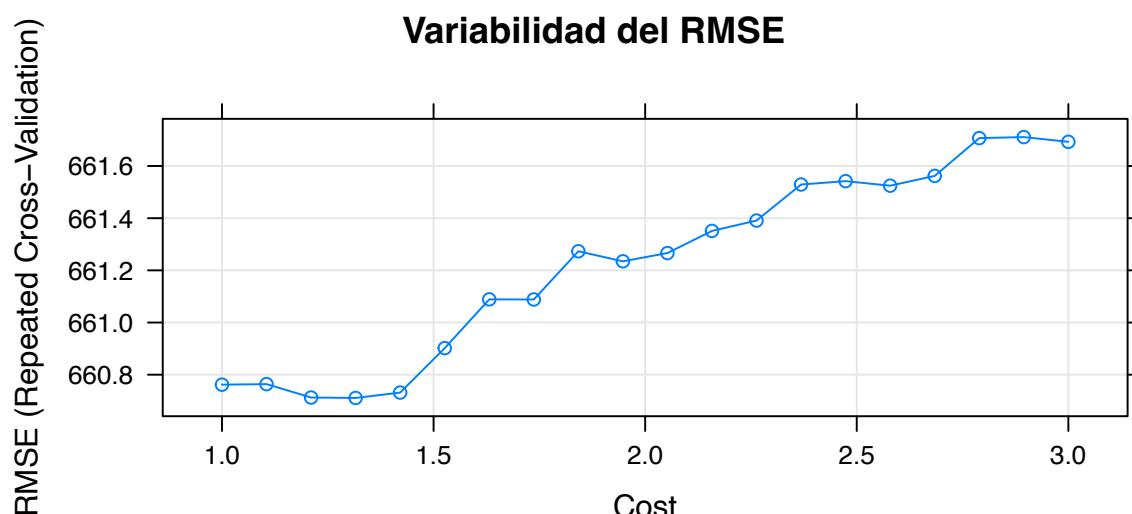
Tabla 5.4: Métricas en el remuestreo. Modelo SVM. Algoritmo 1.1

	RMSE	Rsquared	MAE	Resample
	1064.4454	0.3088627	453.5632	Fold2.Rep1
	1184.6328	0.2829904	458.7337	Fold1.Rep1
	293.4512	0.8920958	223.3695	Fold5.Rep2
	1137.7534	0.3301661	394.2148	Fold2.Rep3
	450.3332	0.7131619	278.7487	Fold4.Rep3
	294.5183	0.8869078	209.1698	Fold3.Rep1
	330.2677	0.8611998	241.7020	Fold5.Rep1
	302.5374	0.8720160	230.8551	Fold2.Rep2
	1586.7542	0.0591160	701.2347	Fold4.Rep2
	1133.8313	0.1815909	529.3938	Fold1.Rep3
	435.7051	0.7702168	268.4395	Fold3.Rep3
	262.8688	0.8897975	200.3649	Fold5.Rep3
	511.0738	0.5204610	295.4428	Fold4.Rep1
	488.7918	0.7056551	266.0491	Fold1.Rep2
	433.6988	0.7351391	251.7757	Fold3.Rep2

Observando los resultados, llegamos lo siguiente:

- El modelo consigue explicar un 60.06 % de la variabilidad total del volumen de ventas para los datos de entrenamiento.
- El error cuadrático medio es de 661 unidades, que es un valor alto para el volumen de ventas que se predice.
- Respecto al remuestreo en la validación cruzada, las métricas no presentan gran variabilidad, en algunas ocasiones se obtiene un R^2 por encima de 0.85 y en otras menor que 0.3, aunque la mayoría de veces se mantiene con un R^2 mayor que 0.7. Por tanto, podemos decir que el modelo es robusto y por tanto, las métricas serán fiables.

En el gráfico mostrado a continuación, se puede observar la variabilidad del error cuadrático medio en función del valor de costo:



A medida que el valor del costo es mayor, el RMSE aumenta considerablemente.

5.2.4.2.2.2. Algoritmo 2: K-Nearest Neighbor Regression (KNN)

Hiperparámetros del algoritmo

- Validación cruzada con 5 grupos y tres repeticiones
- Número de vecinos, k: malla para 3,5,7 y 9

Modelado

```
# Malla para hiperparámetros
KNNGrid <- expand.grid(k = seq(3,9, by=2))

set.seed(17)
modeloKNN_T <- train(VENTAS~.,
                      data = DatosEntreamiento_Total[,-c(1,5,6)],
                      method = "knn",
                      trControl=fitControl,
                      preProcess=c("center","scale"),
                      tuneGrid = KNNGrid)
```

Resultados

El con $K = 7$ vecinos es el que nos proporciona mejores métricas:

Tabla 5.5: Métricas del mejor modelo KNN. Algoritmo 2.1

k	RMSE	Rsquared	MAE	RMSESD	RsquaredSD	MAESD
3 7	692.3719	0.5669368	371.4261	366.764	0.2216798	128.5426

Métricas del remuestreo

Tabla 5.6: Métricas en el remuestreo. Modelo KNN. Algoritmo 2.1

	RMSE	Rsquared	MAE	Resample
1104.5804	0.3741021	496.7094	Fold1.Rep1	
949.8946	0.4590495	439.4187	Fold2.Rep1	
1572.4870	0.0765971	705.3367	Fold4.Rep2	
523.3315	0.6660395	322.7905	Fold1.Rep2	
517.1772	0.6404919	294.8424	Fold3.Rep1	
504.1204	0.6978479	306.0102	Fold3.Rep3	
323.0070	0.8668280	244.0887	Fold5.Rep2	
330.3074	0.8507064	238.5459	Fold2.Rep2	
501.5117	0.5115971	305.9048	Fold4.Rep1	
463.9028	0.6934043	292.8079	Fold4.Rep3	
1025.1152	0.3177601	530.6378	Fold1.Rep3	
587.9297	0.5137595	350.6238	Fold3.Rep2	
370.3185	0.8340566	268.3980	Fold5.Rep1	
514.0675	0.6271062	334.1000	Fold5.Rep3	

1097.8273	0.3747050	441.1762	Fold2.Rep3
-----------	-----------	----------	------------

Observando los resultados, llegamos lo siguiente:

- El modelo consigue explicar un 56.69 % de la variabilidad total del volumen de ventas para los datos de entrenamiento
- El error cuadrático medio es de 661 unidades
- Respecto al remuestreo en la validación cruzada, observamos mayor variación de las métricas, llegando a obtener un R^2 mayor que 0.86, pero obteniendo valores menores que 0.35 en más de una ocasión. La variabilidad indica que el modelo no será tan robusto y por tanto, que las métricas serán menos fiables.

5.2.4.2.2.3. Algoritmo 3: Extreme Gradient Boosting (XGBoost)

Hiperparámetros del algoritmo

- Validación cruzada con 5 grupos y tres repeticiones
- Número de pruebas de hiperparametrización (tune length): 5

Modelado

```
set.seed(17)
modeloXGB_T <- train(VENTAS~.,
                      data = DatosEntreamiento_Total[,-c(1,5,6)],
                      method = "xgbTree",
                      trControl=fitControl,
                      preProcess=c("center","scale"),
                      tuneLength=5, verbosity=0)
```

Resultados

A continuación mostramos la configuración del mejor modelo y las métricas obtenidas:

Tabla 5.7: Métricas del mejor modelo XGBoost. Algoritmo 3.1

eta	max_depth	gamma	colsample_bytree	min_child_weight	subsample	nrounds	RMSE	Rsquared	MAE	RMSD	RsqaredSD	MAESD
131	0.3	3	0	0.8	1	0.625	50	743.499	0.5273148	436.3185	353.9052	0.2188667

Métricas del remuestreo:

Tabla 5.8: Métricas en el remuestreo. Modelo XGBoost. Algoritmo 3.1

RMSE	Rsquared	MAE	Resample
658.1651	0.4534891	453.1551	Fold3.Rep2
310.0093	0.8663871	234.6697	Fold2.Rep2
641.9839	0.6267043	393.1929	Fold3.Rep3
962.6520	0.4458194	436.4683	Fold2.Rep1
435.8219	0.7575645	348.1510	Fold5.Rep2
551.9624	0.4987414	391.8281	Fold4.Rep1
1070.7182	0.2691665	541.5561	Fold1.Rep3
421.4844	0.7818069	309.6371	Fold1.Rep2

Tabla 5.8: Métricas en el remuestreo. Modelo XGBoost. Algoritmo 3.1 (*continued*)

RMSE	Rsquared	MAE	Resample
515.9163	0.6581036	407.1163	Fold4.Rep3
1560.0187	0.1103231	726.9542	Fold4.Rep2
804.5749	0.3864338	496.3289	Fold5.Rep3
1101.5879	0.3596894	536.2408	Fold2.Rep3
1147.4879	0.3254675	614.1487	Fold1.Rep1
557.8004	0.5874277	353.0493	Fold3.Rep1
412.3023	0.7825979	302.2813	Fold5.Rep1

Con estos resultados:

- El mejor modelo consigue explicar un 52.73 % de la variabilidad total del volumen de ventas para los datos de entrenamiento
- El error cuadrático medio es de 743 unidades
- Respecto al remuestreo en la validación cruzada, vuelve a ocurrir como con el segundo modelo, existe mucha variabilidad en las métricas. En ocasiones el modelo predice el volumen de ventas consiguiendo explicar más de un 85 % de la variabilidad total y en otros casos lo hace bastante mal. El coeficiente de determinación varía entre un valor de 0.1103231 y 0.8663871, por lo que las predicciones no tienen mucha fiabilidad.

5.2.4.2.2.4. Prueba de los modelos en los datos de testeo y elección del modelo final

Configuramos los tres modelos con los mejores hiperparámetros y mostramos a continuación una tabla con el coeficiente de determinación y el error cuadrático medio de los tres modelos para poder seleccionar un modelo óptimo que aplicar a los datos de testeo:

Modelo	RMSE	R2
SVM	660.7109	0.6006251
KNN	692.3719	0.5669368
XGBoost	439.0573	0.5220638

Observando la tabla, el modelo seleccionado para predecir el volumen total de ventas en los datos de testeo es el modelo de máquina de vector soporte, ya que las métricas ofrecidas por este modelo son las mejores. Además, era el modelo más robusto, al tener menor variabilidad las métricas en el remuestreo.

Predicción del volumen total de ventas para el conjunto de datos test

En la tabla mostrada a continuación podemos observar las predicciones para los datos de testeo, el valor real y el error absoluto de ventas cometido.

Tabla 5.9: Resultados de predicción. Volumen total de ventas. Algoritmo Máquina de vector soporte

Fecha	Predicción	Valor real	Error absoluto en la predicción
2020-12-24	2064	2090	26
2020-12-26	2675	340	2335
2020-12-27	217	348	131
2020-12-28	2304	3004	700
2020-12-29	2067	2770	703
2020-12-30	2116	2637	521
2020-12-31	2066	2013	53
2021-01-02	2675	2642	33
2021-01-03	231	191	40
2021-01-04	2304	2951	647
2021-01-05	2075	2898	823
2021-01-06	2150	30	2120
2021-01-07	2068	2599	531
2021-01-08	2255	2884	629
2021-01-09	2669	2546	123
2021-01-10	207	161	46
2021-01-11	2307	2492	185
2021-01-12	2064	2496	432
2021-01-13	2127	2116	11
2021-01-14	2072	2058	14
2021-01-15	2248	2607	359
2021-01-16	2669	2859	190
2021-01-17	228	99	129
2021-01-18	2298	2399	101
2021-01-19	2068	2153	85
2021-01-20	2117	1830	287
2021-01-21	2066	2100	34
2021-01-22	2257	2433	176
2021-01-23	2665	2992	327
2021-01-24	215	118	97
2021-01-25	2295	2407	112
2021-01-26	2067	2094	27
2021-01-27	2123	2373	250
2021-01-28	2065	2306	241
2021-01-29	2259	2621	362
2021-01-30	2673	2966	293

```
##          RMSE      Rsquared       MAE
## 624.0146677  0.5860334 365.9166667
```

El modelo *máquina de vector soporte* explica un 58.6 % de la variabilidad total del

volumen de ventas en los datos de testeo. Esta métrica ha mejorado, indicando que el modelo ha sabido generalizar con datos nuevos. El RMSE tiene un valor alto para el volumen de ventas diario, ya que hay un error de unas 624 ventas.

5.2.4.2.3. Predicción del volumen de ventas del producto con calcio

División de los datos en entrenamiento y testeo

Tomamos una partición de 80% 20% para los datos de entrenamiento y testeo:

```
# Datos de entrenamiento
DatosEntreamiento_Calcio <- VolumenVentas_CALCIO[indient,]
# Datos de testeo
DatosTesteo_Calcio<-VolumenVentas_CALCIO[inditest,]
```

5.2.4.2.3.1. Algoritmo 1: Máquina de vector soporte (SVM)

Hiperparámetros del algoritmo

- Validación cruzada con 5 grupos y tres repeticiones
- Parámetro de costo, C: malla de valores entre 1 y 3. Este parámetro penaliza al modelo por cometer errores. Cuanto mayor sea su valor, menos probable es que el algoritmo realice una predicción errónea.

Modelado

```
# Malla para hiperparámetros
# SVMGrid <- expand.grid(C = seq(1,3, length = 20))

set.seed(17)
modeloSVM_C <- train(VENTAS~.,
                      data = DatosEntreamiento_Calcio[,-c(1,5,6)],
                      method = "svmLinear",
                      trControl=fitControl,
                      preProcess=c("center","scale"),
                      tuneGrid = SVMGrid)
```

Resultados

El modelo con un costo $C = 2.4736842$ es el que proporciona mejores métricas:

Tabla 5.10: Métricas del mejor modelo SVM. Algoritmo 1.2

	C	RMSE	Rsquared	MAE	RMSESD	RsquaredSD	MAESD
15	2.473684	410.053	0.5084297	200.9021	242.3671	0.2861008	75.31495

Métricas del remuestreo:

Tabla 5.11: Métricas en el remuestreo. Modelo SVM. Algoritmo 1.2

RMSE	Rsquared	MAE	Resample
605.2194	0.1393749	273.8335	Fold2.Rep1

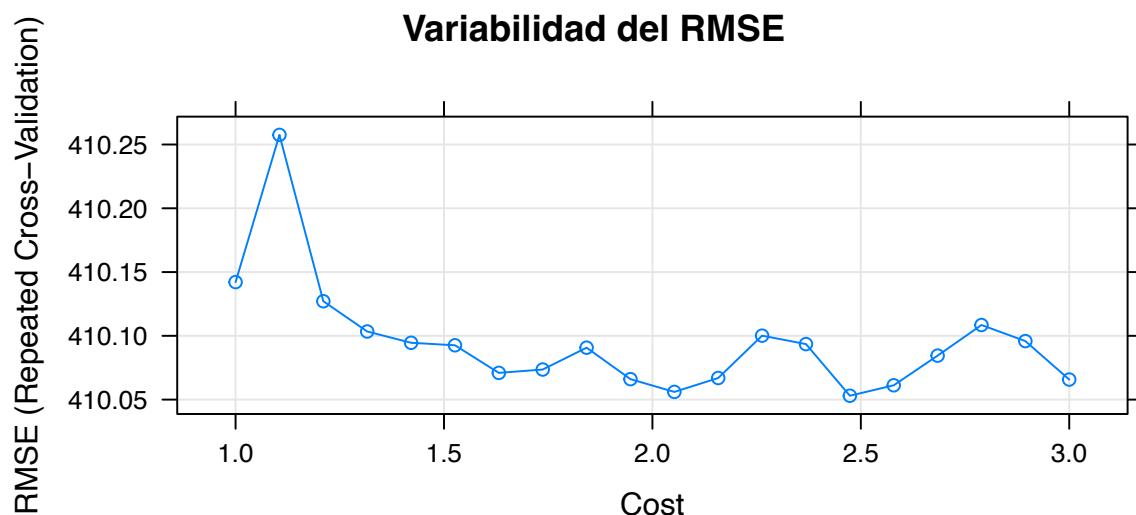
Tabla 5.11: Métricas en el remuestreo. Modelo SVM. Algoritmo 1.2 (*continued*)

	RMSE	Rsquared	MAE	Resample
	262.0413	0.7387940	157.8381	Fold1.Rep1
	251.6186	0.7383878	151.9177	Fold4.Rep3
	695.3437	0.2637021	246.0499	Fold3.Rep1
	224.5364	0.7784103	134.4654	Fold5.Rep1
	747.3853	0.0781575	317.9821	Fold2.Rep2
	741.8805	0.1906661	282.7726	Fold4.Rep2
	316.2545	0.3271437	174.8301	Fold1.Rep3
	237.5538	0.7685589	145.4539	Fold3.Rep3
	225.7057	0.7597007	149.7741	Fold5.Rep3
	273.5356	0.5252290	169.9023	Fold4.Rep1
	238.7011	0.6690615	171.9758	Fold1.Rep2
	209.9762	0.8234289	116.4137	Fold3.Rep2
	250.9025	0.7109831	158.5497	Fold5.Rep2
	870.1407	0.1148477	361.7720	Fold2.Rep3

Observando los resultados, llegamos lo siguiente:

- El modelo consigue explicar un 50.84 % de la variabilidad total del volumen de ventas para los datos de entrenamiento, siendo por tanto el mejor modelo obtenido hasta el momento.
- El error cuadrático medio es de 410 unidades vendidas, que es un valor alto para el volumen de ventas medio diario para productos con calcio.
- Respecto al remuestreo en la validación cruzada, no hay mucha variabilidad en las métricas, en ciertas ocasiones predice muy mal y en otras lo hace algo mejor, aunque en la mayoría de ocasiones el R^2 se mantiene alrededor de un 77 %.

En el gráfico mostrado a continuación, se muestra la variabilidad del error cuadrático medio en función del valor de costo:



Del gráfico podemos concluir que a medida que el costo es mayor, el error cuadrático medio disminuye considerable.

5.2.4.2.3.2. Algoritmo 2: K-Nearest Neighbor Regression (KNN)

Hiperparámetros del algoritmo

- Validación cruzada con 5 grupos y tres repeticiones
- Número de vecinos, k: malla para 3,5,7 y 9

Modelado

```
# Malla para hiperparámetros
# KNNGrid <- expand.grid(k = seq(3,9, by=2))

set.seed(17)
modeloKNN_C <- train(VENTAS~.,
                      data = DatosEntreamiento_Calcio[,-c(1,5,6)],
                      method = "knn",
                      trControl=fitControl,
                      preProcess=c("center","scale"),
                      tuneGrid = KNNGrid)
```

Resultados

El modelo que nos ofrece mejores métricas utiliza 3 vecinos, $K = 9$:

Tabla 5.12: Métricas del mejor modelo KNN. Algoritmo 2.2

k	RMSE	Rsquared	MAE	RMSESD	RsquaredSD	MAESD
4 9	427.7454	0.4770314	234.2056	195.8891	0.2090557	65.06783

Métricas del remuestreo:

Tabla 5.13: Métricas en el remuestreo. Modelo KNN. Algoritmo 2.2

RMSE	Rsquared	MAE	Resample
337.8814	0.6040258	227.8145	Fold1.Rep1
585.0154	0.1880977	325.5011	Fold2.Rep1
888.5876	0.0694378	393.2268	Fold2.Rep3
681.4902	0.3299663	277.9242	Fold4.Rep2
350.2643	0.4001104	245.8416	Fold1.Rep2
658.1906	0.3373078	264.7352	Fold3.Rep1
251.5721	0.7391376	163.6813	Fold3.Rep3
261.4012	0.7086706	160.2794	Fold5.Rep2
545.5286	0.3263769	256.0068	Fold2.Rep2
267.6461	0.5366879	171.8176	Fold4.Rep1
320.6031	0.6221833	192.9137	Fold4.Rep3
365.9266	0.3394912	252.7181	Fold1.Rep3
366.9094	0.5378500	216.6215	Fold3.Rep2
264.6844	0.7083285	179.0464	Fold5.Rep1
270.4791	0.7077993	184.9561	Fold5.Rep3

Observando los resultados, llegamos lo siguiente:

- El mejor modelo consigue explicar un 47.7 % de la variabilidad total del volumen de ventas para los datos de entrenamiento.
- El error cuadrático medio del mejor modelo es de 410, 410, 410, 410, 410 unidades.
- Respecto al remuestreo en la validación cruzada, observamos que las métricas presentan cierta variabilidad, llegando a obtener un R^2 alrededor de 0.739 en alguna ocasión, aunque también se obtienen valores menores que 0.1 varias veces.

5.2.4.2.3.3. Algoritmo 3: Extreme Gradient Boosting (XGBoost)

Hiperparámetros del algoritmo

- Validación cruzada con 5 grupos y tres repeticiones
- Número de pruebas de hiperparametrización (tune length): 5

Modelado

```
set.seed(17)
modeloXGB_C <- train(VENTAS~.,
                      data = DatosEntreamiento_Calcio[,-c(1,5,6)],
                      method = "xgbTree",
                      trControl=fitControl,
                      preProcess=c("center","scale"),
                      tuneLength=5,
                      verbosity=0)
```

Resultados

A continuación mostramos la configuración del mejor modelo y las métricas obtenidas:

Tabla 5.14: Métricas del mejor modelo XGBoost. Algoritmo 3.2

	eta	max_depth	gamma	colsample_bytree	min_child_weight	subsample	nrounds	RMSE	Rsquared	MAE	RMSesd	RsqaredSD	MAesd
269	0.4	1	0	0.6	1	0.875	200	417.2405	0.4994309	254.2198	204.0493	0.2414415	63.16719

Métricas del remuestreo:

Tabla 5.15: Métricas en el remuestreo. Modelo XGBoost. Algoritmo 3.2

	RMSE	Rsquared	MAE	Resample
	663.2970	0.3396104	306.1226	Fold3.Rep1
	436.0698	0.2838283	328.1094	Fold1.Rep2
	550.4014	0.3151969	275.3378	Fold2.Rep2
	227.2224	0.7572851	189.8796	Fold5.Rep3
	572.0858	0.2168972	307.4326	Fold2.Rep1
	405.4052	0.3107259	287.6732	Fold4.Rep1
	690.2166	0.3167046	280.9844	Fold4.Rep2
	259.8125	0.7361796	203.2105	Fold3.Rep2
	254.8398	0.6952834	189.3072	Fold5.Rep2
	225.5202	0.7764179	180.8268	Fold5.Rep1
	326.4058	0.4068394	252.4158	Fold1.Rep3

Tabla 5.15: Métricas en el remuestreo. Modelo XGBoost. Algoritmo 3.2 (*continued*)

RMSE	Rsquared	MAE	Resample
218.4057	0.8112577	175.8185	Fold3.Rep3
298.8358	0.6614095	230.0529	Fold4.Rep3
861.7966	0.1298035	388.9337	Fold2.Rep3
268.2929	0.7340241	217.1924	Fold1.Rep1

Observando los resultados, llegamos lo siguiente:

- El mejor modelo consigue explicar un 49.94 % de la variabilidad total del volumen de ventas para los datos de entrenamiento.
- El error cuadrático medio es de 417 unidades.
- Respecto al remuestreo en la validación cruzada, se trata de un modelo donde obtenemos un valor de R^2 por encima del 65 % en la mayoría de ocasiones. El coeficiente de determinación varía entre un valor de 0.1298035 y 0.8112577, por lo que las predicciones serán más fiables que en el resto de modelos.

5.2.4.2.3.4. Prueba de los modelos en los datos de testeo y elección del modelo final

Configuramos los tres modelos con los mejores hiperparámetros y mostramos a continuación una tabla con el coeficiente de determinación y el error cuadrático medio de los tres modelos para poder seleccionar un modelo óptimo que aplicar a los datos de testeo:

Modelo	RMSE	R2
SVM	410.0530	0.5084297
KNN	427.7454	0.4770314
XGBoost	256.4490	0.4874286

Observando la tabla, el modelo seleccionado para predecir el volumen de ventas para el producto con calcio en los datos de testeo es de nuevo el árbol de regresión XGBoost, ya que, a pesar de no tener el coeficiente de determinación más grande, el valor de RMSE es con diferencia bastante mejor.

Predicción del volumen total de ventas para el conjunto de datos test

Tabla 5.16: Resultados de predicción. Volumen total de ventas. Algoritmo XGBoost

Fecha	Predictión	Valor real	Error absoluto en la predicción
2020-12-24	1643	1106	537
2020-12-26	924	162	762
2020-12-27	219	210	9
2020-12-28	1278	1509	231
2020-12-29	1192	1535	343
2020-12-30	1277	1452	175
2020-12-31	1031	1057	26

Tabla 5.16: Resultados de predicción. Volumen total de ventas. Algoritmo XGBoost
(continued)

Fecha	Predicción	Valor real	Error absoluto en la predicción
2021-01-02	1364	1389	25
2021-01-03	219	86	133
2021-01-04	1373	1512	139
2021-01-05	1785	1465	320
2021-01-06	793	12	781
2021-01-07	1515	1406	109
2021-01-08	740	1537	797
2021-01-09	1102	1391	289
2021-01-10	219	95	124
2021-01-11	1278	1265	13
2021-01-12	623	1443	820
2021-01-13	793	1098	305
2021-01-14	1338	1048	290
2021-01-15	740	1509	769
2021-01-16	2095	1506	589
2021-01-17	219	46	173
2021-01-18	967	1390	423
2021-01-19	1063	1140	77
2021-01-20	664	1044	380
2021-01-21	1031	1120	89
2021-01-22	1223	1279	56
2021-01-23	924	1673	749
2021-01-24	219	67	152
2021-01-25	1373	1389	16
2021-01-26	1158	1160	2
2021-01-27	793	1225	432
2021-01-28	1471	1214	257
2021-01-29	1223	1327	104
2021-01-30	1231	1564	333

```
##          RMSE      Rsquared        MAE
## 397.3646171  0.4607176 300.8055556
```

El modelo *XGBoost* explica un 46.07 % de la variabilidad total del volumen de ventas en los datos de testeо. Esta métrica ha empeorado ligeramente con respecto al entrenamiento, pero es prácticamente la misma.

5.2.4.2.4. Predicción del volumen de ventas del producto sin calcio

División de los datos en entrenamiento y testeо

Se ha tomado una partición de 80 % 20 % para los datos de entrenamiento y testeо:

```
# Datos de entrenamiento
DatosEntreamiento_SinCalcio <- VolumenVentas_SIN_CALCIO[indient,]
# Datos de testeo
DatosTesteo_SinCalcio<-VolumenVentas_SIN_CALCIO[inditest,]
```

5.2.4.2.4.1. Algoritmo 1: Máquina de vector soporte (SVM)

Hiperparámetros del algoritmo

- Validación cruzada con 5 grupos y tres repeticiones
- Parámetro de costo, C: malla de valores entre 1 y 3. Este parámetro penaliza al modelo por cometer errores. Cuanto mayor sea su valor, menos probable es que el algoritmo realice una predicción errónea.

Modelado

```
# Malla para hiperparámetros
# SVMGrid <- expand.grid(C = seq(1,3, length = 20))

set.seed(17)
modeloSVM_SC <- train(VENTAS~.,
                        data = DatosEntreamiento_SinCalcio[,-c(1,5,6)],
                        method = "svmLinear",
                        trControl=fitControl,
                        preProcess=c("center","scale"),
                        tuneGrid = SVMGrid)
```

Resultados

El modelo que nos ofrece mejores métricas tiene un costo, $C = 1.5263158$:

Tabla 5.17: Métricas del mejor modelo SVM. Algoritmo 3.1

	C	RMSE	Rsquared	MAE	RMSESD	RsquaredSD	MAESD
6	1.526316	309.0547	0.5716225	158.5509	131.3478	0.2033014	38.09102

Métricas del remuestreo:

Tabla 5.18: Métricas en el remuestreo. Modelo SVM. Algoritmo 3.1

	RMSE	Rsquared	MAE	Resample
	379.1632	0.4990675	158.99993	Fold2.Rep1
	204.9575	0.7788754	122.75716	Fold1.Rep1
	443.8113	0.4002111	194.01664	Fold1.Rep2
	208.8469	0.7322363	124.64843	Fold3.Rep2
	165.4949	0.8472549	128.02981	Fold5.Rep2
	216.9502	0.7135823	137.11221	Fold2.Rep3
	408.8738	0.4577397	168.00639	Fold4.Rep3
	205.8908	0.7431369	126.97571	Fold3.Rep1

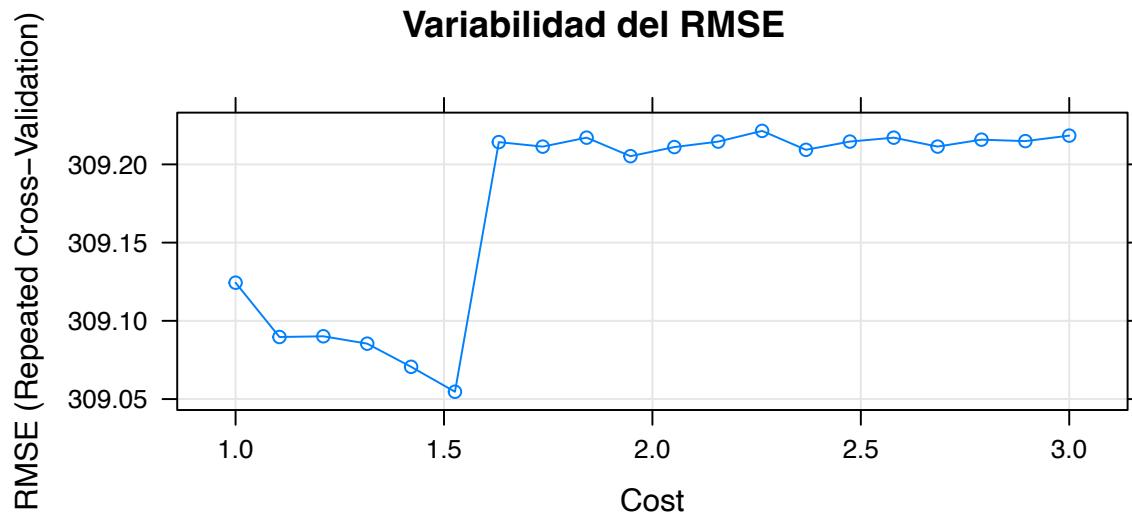
Tabla 5.18: Métricas en el remuestreo. Modelo SVM. Algoritmo 3.1 (*continued*)

	RMSE	Rsquared	MAE	Resample
	229.2722	0.6368688	144.21615	Fold5.Rep1
	471.7089	0.4016428	192.40666	Fold2.Rep2
	274.3370	0.4201040	173.34709	Fold4.Rep2
	118.6151	0.8996526	95.46238	Fold1.Rep3
	297.1549	0.3847775	170.46585	Fold3.Rep3
	491.8929	0.4319479	202.51431	Fold5.Rep3
	518.8506	0.2272394	239.30535	Fold4.Rep1

Observando los resultados, llegamos lo siguiente:

- El modelo consigue explicar un 57.16 % de la variabilidad total del volumen de ventas para los datos de entrenamiento
- El error cuadrático medio es de 309 unidades
- Respecto al remuestreo en la validación cruzada, la variabilidad no es tan evidente como para otros modelos.

En el gráfico mostrado a continuación, se muestra la variabilidad del error cuadrático medio en función del valor de costo:



5.2.4.2.4.2. Algoritmo 2: K-Nearest Neighbor Regression (KNN)

Hiperparámetros del algoritmo

- Validación cruzada con 5 grupos y tres repeticiones
- Número de vecinos, k: malla para 3,5,7 y 9

Modelado

```
# Malla para hiperparámetros
# KNNGrid <- expand.grid(k = seq(3,9, by=2))

set.seed(17)
modeloKNN_SC <- train(VENTAS~,
```

```

data = DatosEntreamiento_SinCalcio[,-c(1,5,6)],
method = "knn",
trControl=fitControl,
preProcess=c("center","scale"),
tuneGrid = KNNGrid)

```

Resultados

El modelo que nos ofrece mejores métricas utiliza 3 vecinos, $K = 9$:

Tabla 5.19: Métricas del mejor modelo KNN. Algoritmo 3.2

k	RMSE	Rsquared	MAE	RMSESD	RsquaredSD	MAESD
4 9	340.3909	0.5055704	197.1153	97.67774	0.1662467	41.19174

Métricas del remuestreo:

Tabla 5.20: Métricas en el remuestreo. Modelo KNN. Algoritmo 3.2

RMSE	Rsquared	MAE	Resample
244.5208	0.7010936	157.9898	Fold1.Rep1
367.7599	0.5235051	191.3800	Fold2.Rep1
324.6113	0.4757384	234.5824	Fold2.Rep3
381.2291	0.3020847	292.0580	Fold4.Rep2
411.8924	0.4975622	192.1033	Fold1.Rep2
247.9909	0.6372968	173.2954	Fold3.Rep1
306.8924	0.3586870	177.7410	Fold3.Rep3
191.9137	0.8136701	140.3458	Fold5.Rep2
445.9473	0.4698573	207.2889	Fold2.Rep2
484.3810	0.3171070	217.0717	Fold4.Rep1
372.8189	0.5442840	184.2665	Fold4.Rep3
180.1500	0.7762747	134.8595	Fold1.Rep3
304.2718	0.4777053	187.1666	Fold3.Rep2
343.0336	0.2817720	220.5716	Fold5.Rep1
498.4504	0.4069177	246.0089	Fold5.Rep3

Observando los resultados, llegamos lo siguiente:

- El mejor modelo consigue explicar un 50.56 % de la variabilidad total del volumen de ventas para los datos de entrenamiento
- El error cuadrático medio del mejor modelo es de 9 unidades
- Respecto al remuestreo en la validación cruzada, observamos que las métricas no presentan una gran variabilidad.

5.2.4.2.4.3. Algoritmo 3: Extreme Gradient Boosting (XGBoost)

Hiperparámetros del algoritmo

- Validación cruzada con 5 grupos y tres repeticiones
- Número de pruebas de hiperparametrización (tune length): 5

Modelado

```
set.seed(17)
modeloXGB_SC <- train(VENTAS~.,
  data = DatosEntreamiento_SinCalcio[,-c(1,5,6)],
  method = "xgbTree",
  trControl=fitControl,
  preProcess=c("center","scale"),
  tuneLength=5,
  verbosity=0)
```

Resultados

A continuación mostramos la configuración del mejor modelo y las métricas obtenidas:

Tabla 5.21: Métricas del mejor modelo XGBoost. Algoritmo 3.3

	eta	max_depth	gamma	colsample_bytree	min_child_weight	subsample	nrounds	RMSE	Rsquared	MAE	RMSESD	RsquaredSD	MAESD
4	0.3	1	0	0.6	1	0.5	200	333.0811	0.5341738	209.154	111.4427	0.1676355	41.15621

Métricas del remuestreo:

Tabla 5.22: Métricas en el remuestreo. Modelo XGBoost. Algoritmo 3.3

	RMSE	Rsquared	MAE	Resample
	378.8303	0.5289871	210.4482	Fold2.Rep1
	264.6700	0.6032428	187.9991	Fold3.Rep2
	322.9133	0.3578732	230.5259	Fold3.Rep3
	372.9588	0.5451704	203.9089	Fold4.Rep3
	136.0411	0.8770442	107.8945	Fold1.Rep3
	325.3041	0.3936659	230.9574	Fold5.Rep1
	236.3685	0.7060187	184.0314	Fold1.Rep1
	291.5687	0.5462843	211.8972	Fold4.Rep2
	507.4771	0.3876681	264.4195	Fold5.Rep3
	501.5461	0.2808295	222.8593	Fold4.Rep1
	396.7435	0.5745140	219.5265	Fold1.Rep2
	206.6731	0.7462074	162.1032	Fold3.Rep1
	502.9724	0.3393894	274.8392	Fold2.Rep2
	251.0204	0.6593545	185.5107	Fold5.Rep2
	301.1296	0.4663568	240.3890	Fold2.Rep3

Observando los resultados, llegamos lo siguiente:

- El mejor modelo consigue explicar un 53.42 % de la variabilidad total del volumen de ventas para los datos de entrenamiento
- El error cuadrático medio es de 333 unidades

- Respecto al remuestreo en la validación cruzada, se trata de un modelo robusto, ya que las métricas no oscilan tanto como en el resto de modelos. El coeficiente de determinación varía entre un valor de 0.2808295 y 0.8770442 por lo que las predicciones serán algo más robustas, a pesar de no tener un valor de R^2 especialmente elevado.

5.2.4.2.4.4. Prueba de los modelos en los datos de testeo y elección del modelo final

Configuramos los tres modelos con los mejores hiperparámetros y mostramos a continuación una tabla con el coeficiente de determinación y el error cuadrático medio de los tres modelos para poder seleccionar un modelo óptimo que aplicar a los datos de testeo:

Modelo	RMSE	R2
SVM	309.0547	0.5716225
KNN	340.3909	0.5055704
XGBoost	211.6134	0.5234797

Observando la tabla, el modelo seleccionado para predecir el volumen de ventas para el producto con sin calcio en los datos de testeo es el modelo de máquina de vector soporte, ya que el valor del coeficiente de correlación es bastante mejor que el resto, a pesar de que el valor de RMSE ofrecido por el modelo XGBoost sea menor.

Predicción del volumen total de ventas para el conjunto de datos test

Tabla 5.23: Resultados de predicción. Volumen total de ventas. Algoritmo máquina de vector soporte

Fecha	Predicción	Valor real	Error absoluto en la predicción
2020-12-24	970	984	14
2020-12-26	1246	178	1068
2020-12-27	104	138	34
2020-12-28	1095	1495	400
2020-12-29	999	1235	236
2020-12-30	984	1185	201
2020-12-31	972	956	16
2021-01-02	1262	1253	9
2021-01-03	104	105	1
2021-01-04	1097	1439	342
2021-01-05	981	1433	452
2021-01-06	979	18	961
2021-01-07	975	1193	218
2021-01-08	1107	1347	240
2021-01-09	1261	1155	106
2021-01-10	104	66	38
2021-01-11	1108	1227	119
2021-01-12	998	1053	55

Tabla 5.23: Resultados de predicción. Volumen total de ventas. Algoritmo máquina de vector soporte (*continued*)

Fecha	Predicción	Valor real	Error absoluto en la predicción
2021-01-13	986	1018	32
2021-01-14	991	1010	19
2021-01-15	1099	1098	1
2021-01-16	1247	1353	106
2021-01-17	104	53	51
2021-01-18	1095	1009	86
2021-01-19	981	1013	32
2021-01-20	979	786	193
2021-01-21	969	980	11
2021-01-22	1089	1154	65
2021-01-23	1246	1319	73
2021-01-24	104	51	53
2021-01-25	1095	1018	77
2021-01-26	981	934	47
2021-01-27	979	1148	169
2021-01-28	970	1092	122
2021-01-29	1128	1294	166
2021-01-30	1263	1402	139

```
##          RMSE       Rsquared        MAE
## 286.2606621  0.6008497 165.3333333
```

El modelo de *máquina de vector soporte* explica un 60.08 % de la variabilidad total del volumen de ventas en los datos de testeo. El resultado obtenido es bastante óptimo y el es el mejor de todo el modelado. El modelo ha sabido generalizar bastante bien con datos nuevos.

5.2.4.2.5. Comparación de resultados del modelado

En la tabla mostrada a continuación se observan las métricas obtenidas trás entrenar los modelos en los correspondientes datasets de testeo:

Tabla 5.24: Resultados de la predicción en los datos de testeo. Comparación de métricas.

Predicción	Modelo	RMSE	Rsquared	MAE
Suma de productos	SVM	624.015	0.586	365.917
Producto con calcio	XGBoost	397.365	0.461	300.806
Producto sin calcio	SVM	286.261	0.601	165.333

Con gran diferencia, el algoritmo que mejor ha sabido generalizar ha sido el correspondiente al producto sin calcio, que explica alrededor del 60 % de la variabilidad total del volumen de ventas, lo cual es buen buen resultado. Para el resto de modelos, no se han obtenido malas métricas, ya que en ambos casos llegamos a explicar más del 45 % de la

variabilidad total, aunque tampoco podemos concluir que el modelo es excelente de cara a predecir el volumen de ventas. Los valores de RMSE también evidencian que el modelo no es bueno, ya que son valores muy altos en comparación con el volumen total de ventas diario correspondiente.

Comparación de la predicción de la suma con la suma de las predicciones

Mostramos un gráfico para comparar la predicción de ventas de la suma de productos con la suma de las predicciones de cada uno de los productos por separado:

Comparación del volumen de ventas



Fuente: Elaboración propia con datos de ventas

Observamos, que de forma *general*, la suma de las predicciones se comporta de manera más acorde a la realidad, es decir, que el volumen de ventas de la suma de las predicciones se parece más al volumen de ventas real. Parece que los modelos, y en especial el modelo de la predicción de la suma, no han sabido captar siempre los valores *extremos*, es decir, cuando se venden muchas unidades (28 de Diciembre) o cuando se venden muy pocas (5 de Enero).

El aspecto que peor han sabido generalizar los modelos es la tendencia al aumento de las ventas para aquellos días donde hubo un mayor volumen de ventas. Por ejemplo, para los días 8, 15 y 27 de Enero, hubo un crecimiento en el número de ventas con respecto al día anterior y según la suma de predicciones, se puede ver como se prevee lo contrario, un descenso de éstas. Lo que si se consigue captar es que el Domingo el volumen de ventas es considerablemente inferior al resto en todos los modelos.

Conclusiones

Por último, en este apartado de conclusiones trataremos de exponer las conclusiones más relevantes del estudio.

Respecto al análisis de cesta de la compra, se ha conseguido obtener una serie de reglas que muestran los patrones de ventas de productos.

Cabe destacar, que la mayoría de ventas consta de uno o dos producto, es decir, no es usual la venta de gran cantidad de productos diferentes en la misma transacción. Otro aspecto a señalar es que se ha detectado un patrón de venta bastante frecuente: el que consta de la venta de los productos 1033 y 1096 de manera conjunta.

Se trata de ventas bastante robustas, ya que los valores del parámetro *lift* son bastante altos, obteniendo en cinco ocasiones un valor de lift por encima de 10.8. Estos resultados indican que las reglas no se deben a la aleatoriedad, sino que es un patrón de comportamiento de ventas real.

En el modelado, se ha encontrado una limitación que ha sido clave a la hora de poder entrenar los modelos, la poca cantidad de días que teníamos. Por este motivo, no se ha podido generar un conjunto de datos para la validación de los modelos, sino que se han generado únicamente conjuntos de entrenamiento y testeo. Esta limitación también ha provocado que los modelos, a pesar de generalizar bien al aplicarlos en los conjuntos de datos de testeo, no hayan conseguido un valor de R^2 cercano a 1 ni un RMSE pequeño en comparación con el volumen de ventas para cada caso.

Además, tampoco se ha podido estudiar la serie temporal de manera mensual o anual, ya que no teníamos datos suficientes.

En general, los modelos obtenidos no consiguen predecir el volumen de ventas con una precisión que podamos considerar aceptable, ya que en el remuestreo observamos que los modelos tampoco eran del todo robustos y presentaban cierta variabilidad. Al trabajar con datos reales, esto es algo que se podía esperar, ya que modelar el comportamiento humano (ventas de productos) no siempre es fácil, ya que se ve afectado por muchos factores, no únicamente precio, día o época del año.

Como posibilidad de mejora, se podría realizar el mismo estudio dentro de unos meses, ya que de esta forma se tendrían mas datos para entrenar y validar los modelos. También podríamos recoger más variables de entrada que pudieran ser de utilidad, como por ejemplo la hora de la venta, las condiciones meteorológicas de las ventas o el lugar donde se ha realizado la transacción, ya que esto arrojaría cierta información adicional que nos podría servir para entender el comportamiento de venta.

Apéndice A

Apéndice: App shiny

Para la exposición del trabajo se ha desarrollado una aplicación web interactiva haciendo uso del paquete de R **Shiny**. Una aplicación Shiny es simplemente un directorio que contiene un script R llamado app.R que se compone de un objeto de interfaz de usuario y una función de servidor.

Se optó por esta opción debido a la amplia gama de opciones que ofrece, diseñando desde cero la propia aplicación, pudiendo insertar gráficos, tablas dinámicas y documentos interactivos. Además, se puede mejorar aún más el rendimiento con acciones JavaScript así como la interfaz de usuario con HTML y el estilo añadiendo código CSS.

La aplicación desarrollada consta de una página inicial y diferentes paneles con las distintas secciones del trabajo, relacionadas con la aplicación práctica para un conjunto datos reales de las técnicas revisadas de forma teórica. Desde la app es fácil navegar por los distintos paneles a través de una barra de navegación.

A continuación se muestran diferentes imágenes de la aplicación:



Figura A.1: Barra de navegación. Fuente: elaboración propia con Shiny.



Figura A.2: Carrusel interactivo. Página de inicio. Fuente: elaboración propia con Shiny.

Contenidos

Análisis de cesta de la compra
Para descubrir asociaciones entre productos.

Aplicación de un proceso de la ciencia de datos
Proceso de limpieza, transformación, depuración y análisis exploratorio de datos.

Predicción. Demanda de productos lácteos
Aplicando modelos estadísticos clásicos y algoritmos de aprendizaje automático.

Análisis de cesta de la compra
Descubriendo asociaciones entre productos

El objetivo principal de este método es identificar patrones o asociaciones entre diversos grupos de productos a partir de las transacciones generadas en cada venta. Estas asociaciones quedan recogidas en las reglas de asociación, agrupaciones de productos en función de su afinidad. La utilidad de las reglas reside en identificar oportunidades de negocio y estrategias para las empresas. Pulse para conocer algunas empresas que utilizan este tipo de análisis. ...

Proceso de ciencia de datos
Aplicado a datos de dos productos lácteos

El objetivo principal de la ciencia de datos consiste en extraer información de valor de los datos. Las etapas de un proceso de ciencia de datos completo son las siguientes:

Pronóstico del volumen de ventas
Basándonos en datos de productos lácteos

A partir de datos de tickets de compra, se han entrenado diferentes modelos para predecir el volumen de ventas de dos productos lácteos por separado y para el volumen de ventas total.

Pulse para ver los modelos entrenados ...

Figura A.3: Índice de contenidos. Página de inicio. Fuente: elaboración propia con Shiny.

Análisis de cesta de la compra con datos de tickets de un supermercado

Se ha aplicado un análisis de cesta de la compra a unos datos que proceden de una muestra de tickets correspondiente a transacciones de una cadena de supermercados. La muestra contiene un total de 7801 tickets que incluyen 4631 artículos distintos.

Conjunto de datos inicial

Show ▾ entries Search:

Idticket	linea	item	cantidad
All	All	All	All
1	1	11802	12
1	2	21662	12
1	3	27959	3
1	4	27958	3
1	5	27950	3
1	6	26574	12
2	1	3334	1
2	2	24347	4
2	3	26785	1
2	4	22498	3

Showing 1 to 10 of 50,000 entries

Previous **1** 2 3 4 5 ... 5000 Next

Conjunto de datos en formato cesta

Show 5 entries Search:

Items
11802,21662,27959,27958,27950,26574
3334,24347,26785,22498,19926,17005,26255,17081,17216,18970,18972,23260,27438,21315,26273,33154,23153,26827,26627
11803,21411,26061
20573,27226,26045,24347,32541,28444,25872,29236,29190,28189,27105,26293,24605,23493,21684,26397,26114,24913,20034,29168
18810,18148,15002,22470,11220,27381
10599,18349,21274,10526,16944,16902,28716
19198,13905
27381,15005
18349
22853,12915,18972,21276,1031,14483,13433,21276,1079

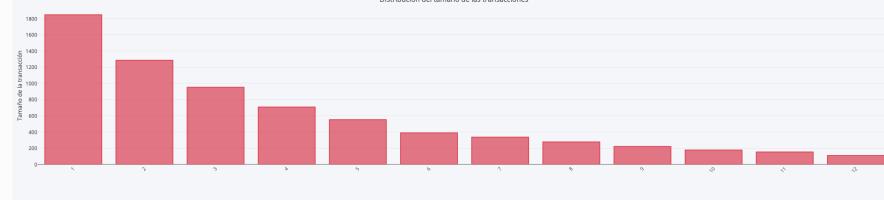
Showing 1 to 10 of 7,801 entries

Previous **1** 2 3 4 5 ... 781 Next

Nota: Únicamente se observa una cota inferior de la venta de cada producto.

Tamaño de las transacciones

Distribución del tamaño de las transacciones



Seleccione el tamaño máximo de transacciones:

 10 19 28 37 46 55 64 73 82

Figura A.4: Página de análisis de cesta de la compra Fuente: elaboración propia con Shiny.

Bibliografía

Xiomara Pamela Silva Cama Aldana Fransheska Dongo Pozo. *Análisis de la minería de datos aplicada en empresas del sector retail*. Facultad de Ingeniería y Computación, Universidad Católica San Pablo, 2020.

Rahul Awad, Marietteand Khanna. *Support Vector Regression*, pages 67–80. Apress, Berkeley, CA, 2015. ISBN 978-1-4302-5990-9. URL https://doi.org/10.1007/978-1-4302-5990-9_4.

Carles Morales Boada. Introducción al feature selection (o cómo escoger las variables adecuadas). Disponible en <https://www.linkedin.com/pulse/introducción-al-feature-selection-o-cómo-escoger-las-morales-boada/?originalSubdomain=es>.

Paula Badia Cambriles. *Análisis cesta de la compra: estudio del método*. Trabajo fin de grado, Departamento de Administración de Empresas y Comercialización e Investigación de Mercados (Marketing) Universidad de Sevilla, 2019.

Andrea Mesta Carmona. *Técnicas estadísticas en predicción de la demanda en el sector comercio*. Trabajo fin de grado, Facultad de Matemáticas, Universidad de Sevilla, 2021.

Guillermo Corres, Alejandra Esteban, Juan García, and Claudia Zárate. Análisis de series temporales. Disponible en <https://economipedia.com/definiciones/sector-retail.html#:~:text=El%20sector%20retail%2C%20o%20comercio,comerciantes%20de%20un%20determinado%20lugar.,01%2009>.

Carlos Alberto Cárdenas Ojeda, Sandra Patricia Ramos Soler. Análisis exploratorio de datos en r, 2015. URL <https://repositorio.uptc.edu.co/handle/001/8178>.

Mario Alcaide Delgado. *Modelo de Regresión Binomial Negativa*. Trabajo fin de grado, Facultad de Matemáticas, Universidad de Sevilla, 2015.

Javier Jesús Espinosa-Zúñiga. Aplicación de algoritmos random forest y xgboost en una base de solicitudes de tarjetas de crédito. *Ingeniería, investigación y tecnología*, 21(3), 2020.

M. Dolores Jiménez Gamero. *Apuntes de la asignatura Series Temporales*, 2021.

Manuel Pérez García. *Técnicas Boosting*. Facultad de Matemáticas, Universidad de Sevilla, 2018.

Cristina Pérez González. *Reglas de asociación. Aplicación práctica en la cesta de la compra de los consumidores.* Trabajo fin de grado, Departamento de Administración de Empresas y Comercialización e Investigación de Mercados (Marketing) Universidad de Sevilla, 2018.

Elena Campo León. *Introducción a las máquinas de vector soporte (SVM) en aprendizaje supervisado.* Facultad de Ciencias, Universidad de Zaragoza.

Pedro L. Luque-Calvo. *Escribir un Trabajo Fin de Estudios con R Markdown,* 2017.

Pedro L. Luque-Calvo. *Cómo crear Tablas de información en R Markdown,* 2019.

Juan M. Muñoz Pichardo Rafael Pino Mejías. *Apuntes de la asignatura Estadística Computacional II,* 2022.

Covadonga Olivera Fernández-Cortés. *Reducción de dimensionalidad en problemas de regresión.* Universidad Carlos III de Madrid. Departamento de Informática, 2017.

Liz et al Pérez-Martínez. Procedimiento para Índice sintético de gestión ambiental: validación con minería de datos. *Ingeniería Industrial,* 42:60 – 87, 08 2021. ISSN 1815-5936.

Luis Quintero Arango. El sector retail, los puntos de venta y el comportamiento de compra de los consumidores de la base de la pirámide en la comuna 10 de la ciudad de medellín. *Revista ciencias estratégicas,* pages 109–118, 2015. URL <https://www.aacademica.org/luis.fernando.quintero.arango/2.pdf>.

Julián Costa Bouzas y Manuel Oviedo del la Fuente Rubén Fernández Casal. *Aprendizaje Estadístico. Bagging,* 2021a.

Julián Costa Bouzas y Manuel Oviedo del la Fuente Rubén Fernández Casal. *Aprendizaje Estadístico. Aprendizaje estadístico vs. Aprendizaje Automático,* 2021b.

Tema 3: Modelos lineales generalizados. Universidad Carlos III de Madrid.

Wikipedia. Walmart. URL <https://es.wikipedia.org/wiki/Walmart>.