Modelo de pronósticos de demanda para la optimización de la cadena de suministros

Aldair Blanco, Angie cantillo

2024-10-03

Resumen

Megatiendas, la cadena de supermercados, se enfrentó a desafíos logísticos significativos en la gestión del suministro para las 25 tiendas que tienen abiertas a la fecha. La falta de una adecuada previsión de la demanda llevó a compras excesivas que no se alinearon con las necesidades del mercado, generando problemas operativos y aumentando el riesgo de pérdida de ventas. La carencia de soporte tecnológico para anticipar de manera efectiva las demandas específicas fue identificada como la raíz de estos desafíos. Con el objetivo de superar estos obstáculos, se implementó un proyecto integral basado en el desarrollo de un modelo de aprendizaje automático. Este modelo buscó prever diariamente la demanda en unidades, centrándose especialmente en la unidad estratégica de negocio FRUVER de Megatiendas. Para llevar a cabo este proyecto, se emplearon herramientas avanzadas como bases de datos en SQL, el lenguaje de programación R y su entorno de desarrollo R Studio. El proceso inició con la creación de un módulo en R diseñado para extraer datos detallados de ventas en unidades de manera precisa, brindando un desglose por tiendas, días, horas y productos específicos. Se realizaron solicitudes formales de acceso a las bases de datos, conexiones VPN y obtención de diccionarios de datos y especificaciones para garantizar la calidad y confiabilidad de la información. Posteriormente, se desarrolló un módulo funcional en R que permitió la transformación, limpieza e integración de datos provenientes de diversas fuentes de Megatiendas. Con el conjunto de datos depurado y listo, se procedió al desarrollo del modelo de aprendizaje automático. Se determinaron las variables relevantes para trabajar y se investigaron diferentes métodos y modelos que se ajustaron a la problemática presentada. Para la evaluación del modelo, se realizó la división del conjunto de datos en un 80% para entrenar y ajustar los modelos, y un 20% para evaluar la capacidad de respuesta de los mismos, siendo este un paso clave. Se llevó a cabo el entrenamiento del modelo con los datos seleccionados y se realizaron las validaciones y ajustes pertinentes para garantizar su eficacia. Este proyecto buscó no solo abordar la problemática actual de Megatiendas en la gestión de inventarios, sino también sentar las bases para una gestión más eficiente y estratégica en el futuro, fortaleciendo la competitividad de la cadena en el mercado. La colaboración con Megatiendas y la aplicación de tecnologías avanzadas en análisis predictivo fueron clave para alcanzar estos objetivos.

Advertencia: Los datos que verá a continuación han sido modificados para preservar los derechos del titular, MEGATIENDAS, de acuerdo con el acuerdo de confidencialidad.

Extracción de datos

En esta etapa inicial, nos enfocamos en la extracción y carga de datos, asegurándonos de que estén actualizados para las siguientes fases de exploración, transformación, y limpieza de datos. Esta fase presentó varios desafíos de seguridad, ya que el cliente MEGATIENDAS rechazó la extracción directa desde nuestro entorno de desarrollo en R mediante VPN, debido a preocupaciones de seguridad. Sin embargo, logramos un acuerdo para crear una cuenta de Google Drive, donde se almacenan los datos en archivos XLSX. Esto garantiza que los datos estén más seguros para ellos y permite a nosotros realizar la extracción de manera eficiente.

El archivo XLSX se actualiza cada primer día del mes con las ventas del mes anterior. Por ejemplo, si estamos en la fecha 30-09-2024, al día siguiente, 01-10-2024, el archivo XLSX se actualizará con las ventas totales de todo septiembre, y así sucesivamente para cada mes. Más adelante en esta etapa, explicaremos cómo se realiza la actualización del archivo y cómo lo integramos a nuestros datos.

Paquete googledrive

El paquete `googledrive` de R es fundamental en nuestro flujo de trabajo debido a la solución adoptada para la extracción y carga de datos desde Google Drive, resultado de las restricciones de seguridad impuestas por MEGATIENDAS. Dado que no se permitió la extracción directa de datos desde nuestro entorno de desarrollo en R mediante VPN, el uso de Google Drive como repositorio de archivos XLSX actualizados mensualmente se convirtió en una opción viable. Este paquete nos permite interactuar con Google Drive desde R, facilitando la automatización del proceso de descarga y carga de archivos de datos.

Con este paquete, podemos:

- **Descargar archivos**: Cada primer día del mes, utilizamos `googledrive` para descargar automáticamente el archivo XLSX actualizado con las ventas del mes anterior.
- **Cargar archivos**: En cada etapa del proyecto, especialmente al finalizar cada una, guardamos los resultados obtenidos en nuestra cuenta de Google Drive. Este enfoque garantiza que todos los avances y resultados intermedios estén organizados y accesibles, permitiendo una continuidad en el flujo de trabajo y asegurando que la información esté siempre disponible para futuras consultas o análisis.
- **Gestión de carpetas**: Aprovechando las funcionalidades de creación y eliminación de carpetas y archivos, optimizamos la organización de nuestros datos y proyectos en Google Drive. Esto nos permite, por ejemplo, asignar una carpeta específica para cada tienda tras la transformación de datos, donde almacenaremos archivos en formato XLSX que serian las entradas para nuestros modelo.

Funciones del Paquete `googledrive` a Utilizar en la Extracción y a lo Largo del Proyecto

El paquete `googledrive` proporciona una serie de funciones útiles para gestionar archivos. A continuación, se detallan algunas de las funciones clave que utilizaremos para la extracción y manejo de datos a lo largo del proyecto:

1. Autenticación y Configuración

- `drive_auth()`: Autentica la conexión con Google Drive, permitiendo a R acceder a los archivos de la cuenta, Además se abrirá una ventana en tu navegador donde podrás ingresar las credenciales de tu cuenta. Una vez autenticado, estas credenciales se guardarán en el entorno de desarrollo, permitiendo un acceso continuo a los recursos sin necesidad de reingresar tus datos cada vez.

2. Gestión de Archivos

- `drive_get()`: Obtiene información sobre archivos específicos en Google Drive. Puedes usar esta función para buscar un archivo por su nombre o ID.
- `drive_download()`: Descarga un archivo a tu entorno local. Esta función es esencial para obtener los archivos XLSX actualizados mensualmente.

- `drive_upload()`: Carga archivos desde tu entorno local. Utilizada para guardar los resultados de los modelos y otros archivos generados.
- `drive_rm()`: Elimina archivos. Puedes usar esta función para gestionar y limpiar archivos antiguos o no necesarios.

3. Gestión de Carpetas

- `drive_mkdir()`: Crea nuevas carpetas. Esta función es útil para organizar los archivos en carpetas específicas, como las correspondientes a cada tienda.
- `drive_find()`: Lista todos los archivos. Permite gestionar y verificar el contenido de tu drive.
- `drive_mv()`: Mueve o renombra archivos y carpetas en Google Drive. Útil para reorganizar la estructura de carpetas según sea necesario.
- `drive_share()`: Comparte archivos o carpetas con otras personas, permitiendo colaborar y compartir resultados con otros miembros del equipo si es necesario.

Una vez comprendidas las funciones disponibles, procederemos a la extracción de datos.

```
drive_auth() #Te abrira una ventana para que te autentiques

drive_find()

file <- drive_get("Consolidados Ventas Fruver x item Enero 2021 a Dic 2023.xlsx")

temp_file <- tempfile(fileext = ".xlsx") # path temporal
drive_download(file, path = temp_file, overwrite = TRUE) # Descarga el archivo a un path temporal
```

Hasta este momento hemos logrado descargar nuestros datos en un archivo temporal de datos, acontinuacion entra en juego una libreria importante para el proyecto.

Librería `readxl`

La librería `readxl` en R es fundamental para trabajar con archivos Excel (.xls y .xlsx). Esta librería facilita la lectura de datos desde hojas de cálculo directamente en R, lo que es crucial cuando los datos están almacenados en este formato.

Funcionalidades Clave de `readxl`

- **Lectura de Archivos Excel**: Permite importar datos desde archivos `.xls` y `.xlsx` en R.
- **Selección de Hojas**: Puedes especificar qué hoja deseas leer si el archivo tiene varias.
- **Manejo de Rangos**: Permite leer rangos específicos de celdas dentro de una hoja de cálculo.
- **Adaptabilidad**: Maneja de manera eficiente diferentes formatos de datos y encabezados.

Funciones Principales

- **`read_excel(path, sheet = NULL, range = NULL, col_names = TRUE, col_types = NULL)`**: Lee los datos desde un archivo Excel. Puedes especificar el archivo, la hoja, el rango de celdas a leer, si incluir nombres de columna, y los tipos de columnas.

Continuaremos con nuestro flujo de trabajo utilizando las funciones de la librería previamente mencionada.

```
DF_202101 <- # Carga la hoja de Excel en un dataframe
  read_excel(temp_file, sheet = "202101") # Lee el archivo Excel directamente desde el path temporal
```

```r
# Cargamos las demas hojas.
DF_202102 <- read_excel(temp_file, sheet = "202102")
DF_202201 <- read_excel(temp_file, sheet = "202201")
DF_202202 <- read_excel(temp_file, sheet = "202202")
DF_202301 <- read_excel(temp_file, sheet = "202301")
DF_202302 <- read_excel(temp_file, sheet = "202302")


# Eliminamos el arhivo temporal
unlink(temp_file)
```

Marco de datos.

Variable	Descripción	Tipo de datos	Valor	Número de observaciones	Número de NAs
Source.Name	Nombre del archivo de origen	character	"ABR 21.txt"	5'214.700	0
Fecha movto.	Fecha del movimiento	datetime	2010-04-01	5'214.700	18
C.O.	Código de operación	numeric	101	5'214.700	18
SECCIONES	Secciones de productos	character	"HORTALIZAS"	5'214.700	18
UNIDAD DE NEGOCIO	Unidad de negocio	character	"FRUVER"	5'214.700	18
CATEGORIAS	Categorías de productos	character	"HOJAS"	5'214.700	18
FAMILIA	Familia de productos	character	"ACELGA"	5'214.700	18
Item	Ítem relacionado	numeric	351652	5'214.700	18
Desc. item	Descripción del ítem	character	"VERDURA ACELGA"	5'214.700	18
Valor subtotal local	Valor subtotal en moneda local	numeric	9189	5'214.700	12
Cantidad inv.	Cantidad inventariada	numeric	3.68	5'214.700	12
Año	Año de los datos	numeric	2010	5'214.700	18
Mes	Mes de los datos	character	"abril"	5'214.700	18
Semana	Semana de los datos	character	"Sem_13"	5'214.700	18

Tabla 1: Marco de datos inicial.

Análisis exploratorio de datos.

Nuestro análisis exploratorio de datos se llevó a cabo utilizando dos herramientas especializadas en el análisis de datos. El análisis que se presenta a continuación fue realizado en R, aprovechando la poderosa librería Shiny, la cual permite crear gráficos interactivos y facilitar la interacción con los usuarios. Sin embargo, debido a las restricciones de protección de datos establecidas en el contrato de confidencialidad del proyecto, solo incluimos imágenes estáticas.

Paralelamente, se realizó un análisis más detallado en Power BI, donde se exploraron muchos más gráficos y se alcanzaron niveles de detalle superiores. Este informe de Power BI fue desarrollado exclusivamente para el cliente MEGATIENDAS y para los involucrados en el desarrollo del proyecto.

Análisis Exploratorio de unidades vendidas con Filtros Clave y Enfoque en Miércoles de Fruver.

Para el primer gráfico de nuestro dashboard desarrollado en Shiny, creamos un gráfico de líneas que permite visualizar el comportamiento de los datos. En el eje Y se muestran las unidades vendidas sumadas, mientras que en el eje X se presentan las fechas de las ventas en formato "nombre del día + número del día del mes". Además, añadimos un panel de filtros clave que permite analizar los datos por regional, tienda, año y mes, facilitando así un análisis exploratorio de datos más preciso.

Este enfoque nos ayudó a identificar anomalías en nuestras series de tiempo. MEGATIENDAS nos solicitó que prestáramos especial atención a los "Miércoles de Fruver", los cuales están señalados en el gráfico mediante una línea que atraviesa todo el gráfico, destacando esta dinámica comercial.

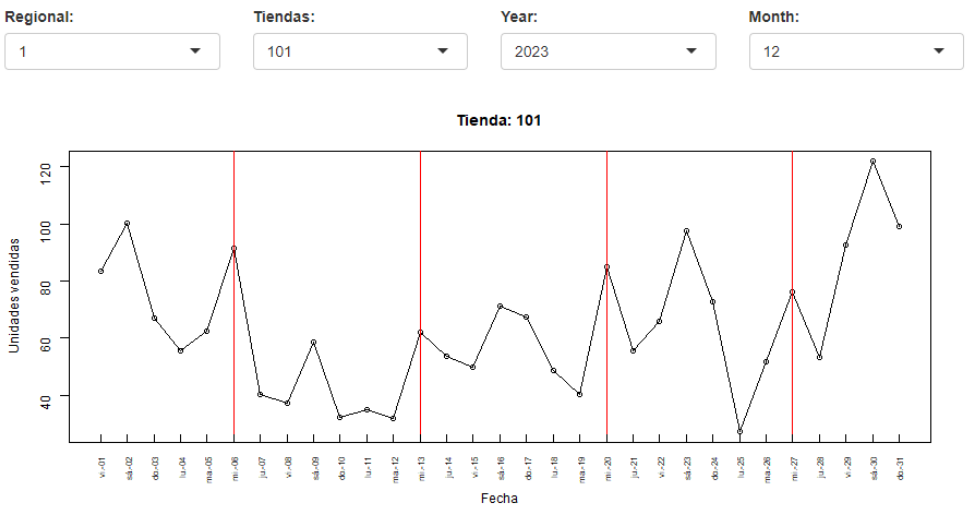


Figura 1: Tendencia de datos para la regional 1, tienda 101, en diciembre de 2023.

Análisis de Tendencias Mensuales por Sección.

Para el segundo gráfico, nos enfocamos en uno de los aspectos fundamentales para MEGATIENDAS: la categorización de los productos a través de sus múltiples secciones, divididas en seis categorías principales. Al analizar nuestros datos de tendencia mensual, observamos que la sección 027, resaltada en azul, no presenta ventas continuas a lo largo de todos los meses del año, registrando ventas únicamente entre febrero y julio para la regional 1, tienda 101.

Asimismo, identificamos que la sección 0060 tiene ventas únicamente entre agosto y diciembre en todos los años analizados. Por otro lado, las secciones 0066, 0061 y 0062 se mantienen consistentemente en el top 3

de ventas durante todos los años de los datos evaluados.

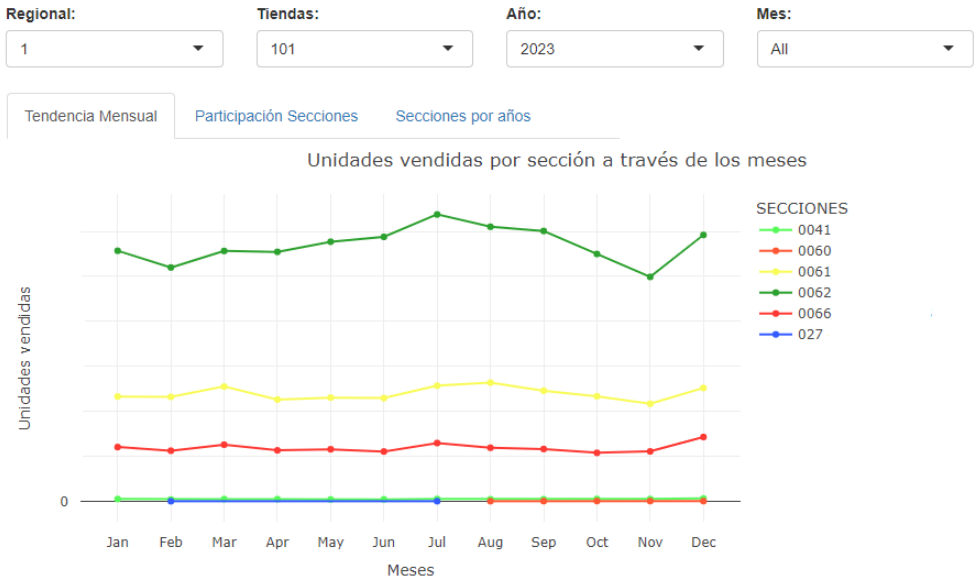


Figura 2: Ventas mensuales por sección: secciones 027 y 0060 estacionales; 0066, 0061, 0062 consistentemente en el top 3 anual.

Distribución de unidades vendidas por Sección.

En el tercer gráfico, nos enfocamos nuevamente en la categorización por secciones de los productos de MEGATIENDAS, esta vez analizando el porcentaje de participación de cada sección a través de un gráfico de torta. Los resultados confirman que las secciones 0062 (61.1%), 0061 (25.7%), y 0066 (12.7%) forman el top 3 de las secciones más vendidas durante el último año de ventas. Estas tres secciones juntas representan el 99.5% de todas las unidades vendidas en la regional 1, tienda 101.

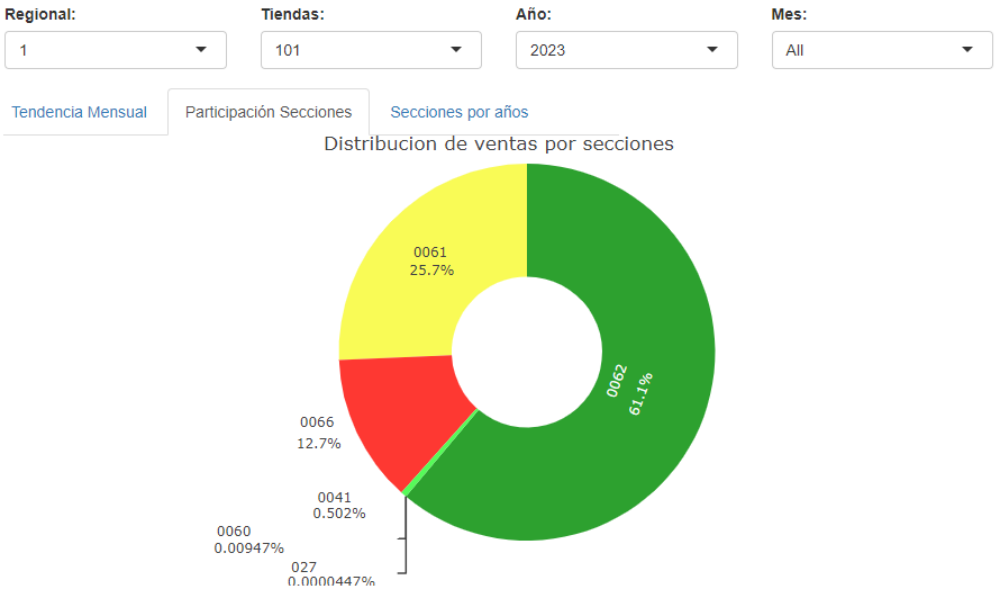


Figura 3: Porcentaje de participación de las secciones en la regional 1, tienda 101, en el año 2023.

Transformación y limpieza de datos.

En esta segunda etapa, centraremos nuestros esfuerzos en la limpieza exhaustiva de nuestros datos, un paso crucial para asegurar la precisión de los análisis posteriores. Comenzaremos identificando y excluyendo las ventas al por mayor, ya que estos datos pueden distorsionar los patrones subyacentes que queremos analizar en nuestro modelo. La exclusión de estas ventas es esencial para garantizar que el conjunto de datos refleje fielmente las condiciones de venta normales, lo que es fundamental para un pronóstico confiable de la serie de tiempo.

Además, abordaremos la eliminación de las devoluciones. Este aspecto es igualmente crítico, ya que las devoluciones pueden introducir ruido y sesgos en el modelo predictivo. Al excluir cuidadosamente estas transacciones, nos aseguramos de que los datos restantes representen con precisión las tendencias y comportamientos reales del mercado, lo cual es indispensable para el desarrollo de un modelo de pronóstico robusto.

Una vez completadas estas tareas de limpieza, generaremos un dataframe final que estará optimizado y listo para ser utilizado en la siguiente fase: Desarrollo del modelo predictivo. Todo el proceso descrito será detallado a continuación, explicando cada paso para que se entienda la importancia y el impacto de estas acciones en la calidad del modelo final.

Eliminar valores NAS.

En el marco de datos presentado inicialmente en la Tabla 1, observamos que la última columna, "Número de NAs", muestra la cantidad de valores faltantes (NAs) en cada columna. Estos NAs son muy escasos, representando menos del 0.01% del total de los datos. Por lo tanto, procederemos a eliminarlos.

Seguimos con nuestro DataFrame obtenido atravez de nuestra extraccion de datos "Ventas_2021_2024", el cual hasta este momento consta de 5'214.700 observaciones.

```
VentasSinMayoristas <- Ventas_2021_2024[!is.na(Ventas_2021_2024$`C.O.`), ]
```

Identificar y excluir ventas al por mayor.

En los supermercados Megatiendas, se consideran ventas mayoristas aquellas en las que un solo comprador adquiere una cantidad máxima de un solo producto en una única factura. Estas ventas deben ser excluidas en esta fase, ya que no son muy comunes y son más bien oportunidades excepcionales. De hecho, existen políticas en Megatiendas que prohíben estas ventas. Sin embargo, las tiendas a veces las realizan para evitar que los productos se desperdicien. Para nosotros, estas ventas irregulares podrían causar distorsiones en nuestros modelos de pronóstico. Por ejemplo, en un modelo de pronóstico de demanda, la inclusión de estas ventas mayoristas podría resultar en previsiones inexactas, afectando la precisión y efectividad del modelo. Además, esto podría contribuir a sobrecompras de productos, generando un exceso de inventario y potenciales pérdidas financieras.

Actualmente, estas ventas mayoristas están presentes en nuestros datos y se distinguen de las demás tiendas por tener un centro de operación (C.O.) igual a 1. En nuestro DataFrame final, debemos filtrar solo las tiendas de las regionales 1 y 2 (C.O < 300), excluyendo aquellas con centro de operación 1 (C.O !=1).

```
VentasSinMayoristas <- VentasSinMayoristas %>%
  filter(C.O. <= 300 & C.O. != 1)
```

Tras el proceso anterior, los datos del dataframe original "Ventas_2021_2024" con 5'214.700 observaciones se reducen a un nuevo dataframe "VentasSinMayoristas" con 4'656.668 observaciones.

Tratamiento devoluciones.

Las devoluciones, como su nombre lo indica, se refieren a productos que el cliente compró y luego decidió regresar a la tienda. Este proceso puede ocurrir por diversas razones. Una de las más comunes es que, al revisar su compra, el cliente descubre que el producto está en estado de descomposición. Sin embargo, las devoluciones también pueden ocurrir por otros motivos, En estos casos, el cliente tiene la opción de devolver el producto a la tienda, el cliente puede elegir entre varias opciones. Puede recibir un reembolso del dinero pagado, cambiar el producto por otro de la misma categoría de productos, o incluso optar por un producto de una categoría diferente, siempre y cuando el precio sea el mismo.

Las devoluciones no están específicamente marcadas en los datos enviados, lo que puede dificultar su identificación directa. Sin embargo, existe un patrón que nos permite reconocerlas: **las devoluciones suelen aparecer con una cantidad vendida en negativo. Esto indica que el producto ha sido retornado y no representa una venta regular.**

En la siguiente sección de código, abordaremos cómo identificar estas devoluciones utilizando esta característica distintiva. Analizaremos los datos para encontrar entradas con cantidades negativas y, posteriormente, las clasificaremos como devoluciones. Dependiendo del porcentaje de datos que representen las devoluciones, se determinará cómo manejar estos datos.

Si las devoluciones no constituyen la mayoría de los datos, es decir, si están por debajo del 10% del total de los datos, procederemos con su eliminación. Esta estrategia nos permitirá mantener la precisión del análisis de ventas y evitar que las devoluciones influyan desproporcionadamente en los resultados. En cambio, si las devoluciones representan un porcentaje significativo, se implementará un tratamiento especial para integrarlas adecuadamente en el análisis.

Validemos el porcentaje de las devoluciones:

```
PorcDevoluciones <- ((count(VentasSinMayoristas[VentasSinMayoristas$`Cantidad inv.` < 0,])/
  count(VentasSinMayoristas))*100)

paste0("Porcentaje de devoluciones ",round(PorcDevoluciones,3),"%")
```

```
## [1] "Porcentaje de devoluciones 0.039%"
```

El número de registros de devoluciones asciende a aproximadamente 1.839 filas de datos, lo que representa un 0.039% del total del DataFrame (4'656.668). Este porcentaje indica que las devoluciones son una parte pequeña del conjunto de datos.

Dado que hemos establecido un umbral del 10% para decidir si se debe dar un tratamiento especial a las devoluciones en nuestro análisis, y considerando que estas devoluciones no alcanzan dicho umbral, hemos decidido proceder con la eliminación de todos estos registros del DataFrame.

```
VentasNoMayoristasNoDevoluciones <- VentasSinMayoristas[VentasSinMayoristas$`Cantidad inv.` >= 0,]
```

Cambio de nombres de variables.

Para facilitar el manejo de las variables en las próximas etapas del proyecto, realizamos cambios fundamentales en los nombres de las columnas de nuestro dataframe "VentasNoMayoristasNoDevoluciones". Es importante destacar que solo se modificarán los nombres de las columnas, sin alterar ninguno de los datos.

```
VentasNoMayoristasNoDevoluciones <- VentasNoMayoristasNoDevoluciones %>%
  rename("origen" = `Source.Name`,
         "fecha" = `Fecha movto.`,
         "centro_op" = `C.O.`,
```

```
        "desc_centro_op" = `Desc. C.O.`,
        "secciones" = SECCIONES,
        "unidad_negocio" = `UNIDAD DE NEGOCIO`,
        "categorias" = CATEGORIAS,
        "familia" = FAMILIA,
        "item" = Item,
        "desc_item" = `Desc. item`,
        "Valor" = `Valor subtotal local`,
        "unidades" = `Cantidad inv.`,
        "año" = Año,
        "mes" = Mes,
        "semana" = Semana)
```

Con los cambios de nombres realizados, nuestro dataframe "VentasNoMayoristasNoDevoluciones" quedaría de la siguiente forma:

Nombres Anteriores	Nombres Nuevos
Source.Name	origen
Fecha movto.	fecha
C.O.	centro_op
Desc. C.O.	desc_centro_op
SECCIONES	secciones
UNIDAD DE NEGOCIO	unidad_negocio
CATEGORIAS	categorias
FAMILIA	familia
Item	item
Desc. item	desc_item
Valor subtotal local	Valor
Cantidad inv.	unidades
Año	año
Mes	mes
Semana	semana

Tabla 2: Renombre de variables.

Guardamos nuestro DataFrame final en drive.

```
### Primero en formato CSV
temp_file <- tempfile(fileext = ".csv")
write.csv(VentasNoMayoristasNoDevoluciones, temp_file, row.names = FALSE)

carpeta <- drive_get("ExtracciónDeDatos")
carpeta_id <- carpeta$id

# Cambiar tiempo de espera al subir el archivo
httr::set_config(httr::config(connecttimeout = 60))

# Subir el archivo temporal a la carpeta específica en Google Drive
drive_upload(media = temp_file, path = as_id(carpeta_id), name = "Ventas_2021_2024.csv")

# Eliminar el archivo temporal si ya no lo necesitas
unlink(temp_file)
```

Generación de DataFrame Final.

En los pasos anteriores, excluimos todas las devoluciones y construimos el DataFrame "VentasSinMayoristas", que contenía 4'656.668 observaciones de las 5'214.700 originales del dataframe "Ventas_2021_2024". Posteriormente, este también se procesó para excluir devoluciones, generando el dataframe "VentasNoMayoristasNoDevoluciones", con 4'654.829 observaciones, el cual en el paso anterior lo guardamos en nuestro drive. Este es nuestro DataFrame final, listo para el próximo objetivo.

Desarrollo del modelo predictivo.

En este módulo, realizaremos una exhaustiva investigación del estado del arte para identificar modelos de aprendizaje automático que se adapten teóricamente a nuestra problemática. Nos centraremos en seleccionar cuatro modelos prometedores basados en su rendimiento teórico y aplicabilidad. Posteriormente, implementaremos estos modelos utilizando nuestros datos históricos de ventas desagregados por fecha, tiendas y productos.

Realizaremos pruebas rigurosas para evaluar el desempeño de cada modelo en la predicción diaria de la demanda en unidades de FRUVER en Megatiendas. Para asegurar la validez de nuestras conclusiones, diseñaremos un esquema de validación robusto que nos permita comparar objetivamente la precisión de cada modelo. Este enfoque nos permitirá determinar cuál de los modelos, o qué combinación de ellos, ofrece la mejor precisión en el pronóstico de la demanda, asegurando así una mejora significativa respecto a los modelos existentes.

Selección de variables.

Como se indicó anteriormente, en esta sección nos centraremos en la selección de las variables predictoras. Al inicio de este documento, se presenta un diccionario de datos que ha sido transformado y depurado a lo largo del proyecto, preservando su estructura original. A partir de todas estas transformaciones almacenadas en el dataframe VentasNoMayoristasNoDevoluciones, procederemos a seleccionar nuestras variables predictoras para el análisis.

Variable entrada	Descripción	Tipo de datos	Valor	Número de observaciones
fecha	Fecha del movimiento de ventas.	date	2021-04-01	4'654.829
centro_op	Código de tiendas.	numeric	101	4'654.829
item	Codigo unico del producto.	numeric	351610	4'654.829
unidades	Cantidad vendida.	numeric	3.68	4'654.829

Tabla 3: Selección de variables.

Seleccionando variables:

```
UnidadesVendidas <- VentasNoMayoristasNoDevoluciones[,c("fecha","centro_op","item","unidades")]
```

Por el momento solo seleccionaremos estas variables, debido a que estamos seguros que algunos modelos requeriran muchas otras variables.

Se evidencia un total de 4'654.829 registros. Estos registros provienen desagregados del dataframe original, lo cual podría causar confusión en nuestros modelos. Dado que la información corresponde a la sumatoria de las unidades vendidas por una tienda en una fecha determinada, procederemos a agrupar el dataframe

UnidadesVendidas por fecha, centro_op, item, utilizando la variable unidades como agrupadora. Nuestro objetivo es reducir y consolidar el dataframe de manera que no haya registros duplicados de ventas de un mismo item en una misma fecha y tienda.

```
UnidadesVendidas <- UnidadesVendidas %>%
                    group_by(.dots=c("fecha","centro_op","item")) %>%
                    summarize(unidades=sum(unidades), .groups = 'drop')

paste0("Total de registros: ",count(UnidadesVendidas))
```

```
## [1] "Total de registros: 4654829"
```

Hemos obtenido un total de 4'654.829 registros, igual a lo que teníamos anteriormente. Sin embargo, con este paso nos aseguramos de que nuestros datos estén perfectamente agrupados y listos para su uso en nuestros modelos.

Clusterización de tiendas.

Antes del desarrollo de los modelos predictivos, es fundamental realizar la clusterización de tiendas. Esto nos permitirá identificar similitudes entre las tiendas, facilitando el agrupamiento y el tratamiento del desarrollo del modelo por bloques.

Proponemos realizar el agrupamiento de tiendas utilizando la medida de similitud más utilizada actualmente, conocida como la distorsión del tiempo o, por su acrónimo en inglés, DTW (Dynamic Time Warping). Esta medida permite comparar series temporales de distinta longitud y localizar similitudes entre ellas, ajustando las discrepancias en el tiempo.

Existen diversas técnicas para comparar las similitudes de series temporales, siendo la DTW y la distancia euclidiana dos de las más comunes. La distancia euclidiana mide la diferencia directa entre puntos correspondientes en dos series temporales, lo que puede ser útil cuando las series están perfectamente alineadas en el tiempo. Sin embargo, esta técnica no es adecuada cuando hay variaciones en la velocidad o en el tiempo de los eventos en las series.

La DTW, en cambio, es capaz de ajustar estas discrepancias temporales al permitir que ciertos puntos de una serie se alineen con múltiples puntos de otra. Esto hace que la DTW sea especialmente útil para comparar series temporales que tienen patrones similares pero no están perfectamente sincronizadas. Por ejemplo, si dos tiendas tienen picos de ventas en diferentes momentos del día o de la semana, la DTW puede identificar estos picos como similares, mientras que la distancia euclidiana podría considerarlos diferentes debido a la desalineación temporal.

Por estas razones, la DTW es una mejor opción para nuestro análisis, ya que proporciona una comparación más flexible y precisa entre las series temporales de las tiendas, permitiendo un agrupamiento más significativo y útil para el desarrollo de modelos predictivos personalizados para cada grupo.

En la siguiente tabla veremos las ventas y desventjas de DTW y Distancia Euclidiana.

Característica	DTW (Dynamic Time Warping)	Distancia Euclidiana
Ventajas		
Maneja discrepancias temporales	Sí	No
Comparación flexible	Sí	No
Identifica patrones similares	Sí, incluso si no están sincronizados en el tiempo	Solo si están perfectamente alineados

Característica	DTW (Dynamic Time Warping)	Distancia Euclidiana
Adecuado para series de distinta longitud	Sí	No, requiere series de igual longitud
Aplicaciones	Útil en reconocimiento de patrones y series temporales no alineadas	Útil en análisis de datos alineados y métricas simples
Desventajas		
Complejidad computacional	Alta, debido a la necesidad de calcular alineaciones óptimas	Baja, cálculo directo
Sensibilidad al ruido	Puede ser sensible al ruido, aunque menos que la distancia euclidiana si se aplica suavizado	Alta, puede ser muy sensible al ruido y a pequeñas variaciones
Interpretación	Más compleja, requiere comprensión de la alineación temporal	Simple, diferencia directa entre puntos correspondientes
Necesidad de preprocesamiento	Puede requerir preprocesamiento para reducir ruido y optimizar la alineación	Generalmente menos preprocesamiento necesario

Tabla 4: Ventajas y desventajas de DTW Vs Distancia Euclidiana.

Análisis con DTW.

1 - Agrupamos las unidades vendidas por tienda y fecha. Este paso es crucial porque la estructura de nuestro DataFrame final, tras la selección de variables, incluye la fecha, el código de tienda, el artículo y la cantidad de inventario. Por lo tanto, para analizar la similitud entre las tiendas, es absolutamente necesario agrupar los datos por tienda, creando un nuevo DataFrame, diferente del DataFrame final, en el que conservaremos la estructura de fecha, código de tienda y la suma de las cantidades de inventario.

```
#Agrupamos el DataFrame VentasNoMayoristasNoDevoluciones, previamente limpiado en los pasos anteriores,

DtwTiendas <- UnidadesVendidas %>%
                  group_by(.dots=c("fecha","centro_op")) %>%
                   summarize(Unidades = sum(unidades, na.rm = TRUE), .groups = "drop")

colnames(DtwTiendas)[1] <- "Fecha"
colnames(DtwTiendas)[2] <- "Tiendas"
```

2 - Pivoteamos el DataFrame para reorganizar los datos. Estructurar los datos con fechas como columnas y tiendas como filas permite representar cada tienda como una serie temporal independiente, facilitando la comparación de similitudes entre tiendas mediante DTW. Esto es esencial para calcular las similitudes, realizar clustering y mejorar la interpretabilidad y visualización de los resultados, ya que este formato es ideal para analizar y comparar el comportamiento de las tiendas a lo largo del tiempo.

```
# Pivotar el DataFrame para tener fechas como columnas y tiendas como filas
 DtwTiendas_pivot <- DtwTiendas %>%
  pivot_wider(names_from = Fecha, values_from = Unidades, values_fill = list(Unidades = 0))
```

3 - Eliminamos la columna de tiendas para crear la matriz de datos. Convertir la columna de tiendas en nombres de filas y luego transformar los datos a una matriz tiene como objetivo estructurarlos de manera que sean directamente utilizables por el algoritmo DTW. Esta organización optimiza la ejecución de dichos algoritmos, asegurando que cada serie temporal esté correctamente identificada y asociada con su tienda correspondiente.

```
# Convertir la columna 'Tiendas' en los nombres de las filas y eliminar la columna
df_rownames <- DtwTiendas_pivot %>%
```

```
  remove_rownames() %>%
  column_to_rownames(var = "Tiendas")

# Convertir a matriz
df_matrix <- as.matrix(df_rownames)
```

4 - Matriz de distancia y aplicación del algoritmo DTW. Calculamos la matriz de distancias entre cada par de tiendas utilizando la métrica DTW, que mide la distancia entre sus series temporales permitiendo desalineaciones en el tiempo. A partir de esta matriz de distancias, aplicamos el algoritmo de clustering jerárquico para agrupar las tiendas según la similitud de sus series temporales.

```
ClusteresTiendas <- hclust(dist(df_matrix, method = "dtw"))
```

5 - Guardamos los resultados obtenidos. Se calcula la asignación de 4 clusters para cada tienda, se añade esta información al DataFrame original para facilitar la visualización y se crea un nuevo DataFrame ClusteringTiendas con los nombres de las tiendas y sus números de cluster, útil para análisis y visualización.

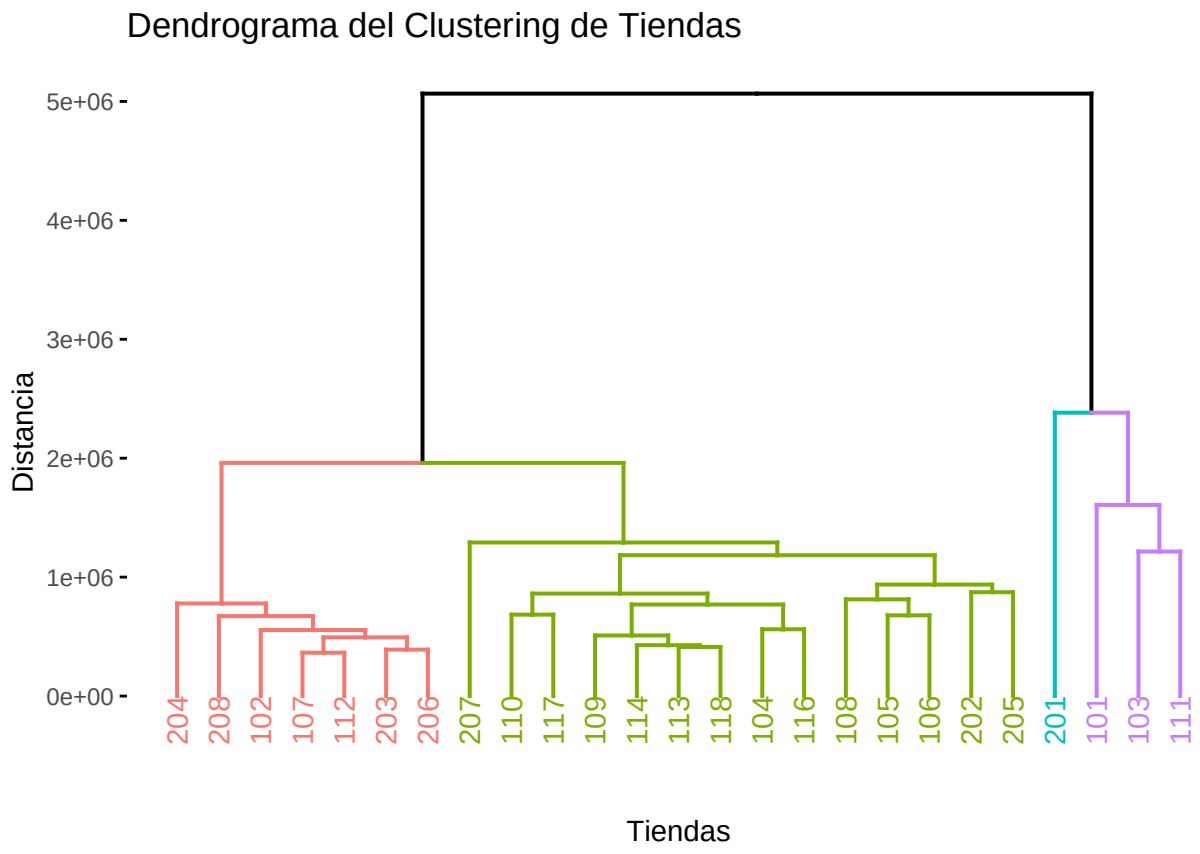
```
ClusteringTiendas <- cutree(ClusteresTiendas,k=4)

DtwTiendas_pivot <- cbind(DtwTiendas_pivot,ClusteringTiendas)

ClusteringTiendas <- as.data.frame( cbind(DtwTiendas_pivot$Tiendas,DtwTiendas_pivot$ClusteringTiendas)
)
```

6 - Visualizamos los Clusters. Generamos un dendrograma utilizando la función fviz_dend del paquete factoextra, que permite visualizar cómo se agrupan las observaciones en diferentes clusters. La línea horizontal en el gráfico indica el nivel de corte utilizado para formar los clusters.

```
fviz_dend(ClusteresTiendas, k = 4,
          main = "Dendrograma del Clustering de Tiendas",
          xlab = "Tiendas",
          ylab = "Distancia") +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank())
```



El análisis de similitud de series temporales mediante el uso del algoritmo DTW fue altamente efectivo, permitiendo agrupar las tiendas en 4 clusters distintos. Estos clusters se formaron en función de las similitudes identificadas en las series temporales de unidades vendidas en cada tienda, lo que facilita una comprensión más profunda de los patrones de comportamiento entre los diferentes grupos. Este proceso no solo permitió categorizar las tiendas según sus patrones de unidades vendidas a lo largo del tiempo, sino que también sienta las bases para la estrategia que desarrollaremos a continuación.

Muestra intencionada basada en la clasificación de unidades vendias.

Para el desarrollo del pronóstico de unidades vendidas, seleccionaremos una tienda representativa de cada cluster. Dentro de cada una de las tiendas, se elegirá una muestra representativa de los productos con más unidades vendidas. De un total de aproximadamente 150 productos por tienda, se seleccionarán los 30 productos con mayores unidades vendidas. Dentro de este grupo, se hará una selección intencionada para garantizar la inclusión de productos con diferentes niveles de ventas: los dos productos más vendidos, dos productos con ventas intermedias (posiciones 15 y 16), y los dos productos con las ventas más bajas dentro del Top 30 (posiciones 29 y 30). Este enfoque no solo permitirá evaluar el desempeño de los modelos de pronóstico en productos con distintos niveles de demanda, desde los más populares hasta los menos vendidos, sino que también facilitará una retroalimentación rápida, lo que nos ahorrará tiempo, permitiéndonos enfocarnos en los ajustes necesarios para optimizar nuestros modelos de pronóstico.

Para llevar a cabo el procedimiento anterior, se desarrolló una función en R denominada **Funct_TopItems.** A continuación, se presentará la función en su totalidad, seguida de una explicación detallada de cada una de sus partes y su respectiva funcionalidad.

```r
Funct_TopItems <- function(Tienda) {

    primer_dia_hace_dos_meses <- floor_date(max(UnidadesVendidas$fecha) %m-% months(1), "month")

    DatosTienda <- UnidadesVendidas[UnidadesVendidas$centro_op == Tienda &
                        UnidadesVendidas$fecha >= primer_dia_hace_dos_meses,]

    # Agrupar por item y sumar las unidades
    top_items <- DatosTienda %>%
      group_by(item) %>%
      summarize(total_unidades = sum(unidades)) %>%
      arrange(desc(total_unidades))  %>%
      head(30)

    # Graficar
    p <- ggplot(top_items, aes(x = reorder(item, -total_unidades), y = total_unidades,
                               fill = total_unidades)) +
      geom_bar(stat = "identity") +
      scale_fill_gradient(low = "lightblue", high = "steelblue") +
      scale_y_continuous(labels = comma) +
      labs(title = paste0("Top 30 Items Más Vendidos, Tienda: ",Tienda), x = "Item",
           y = "Total Unidades Vendidas") +
      theme_minimal() +
      theme(
        legend.position = "none",
        axis.text.x = element_text(angle = 45, hjust = 1),
        axis.text.y = element_blank()
      ) +
      annotate("rect", xmin = 0.5, xmax = 2.5,
               ymin = 0, ymax = max(top_items$total_unidades), color = "red",
               fill = NA, size = 1.2) +
      annotate("rect", xmin = 13.5, xmax = 15.5,
               ymin = 0, ymax = max(top_items$total_unidades), color = "red",
               fill = NA, size = 1.2) +
      annotate("rect", xmin = 28.5, xmax = 30.5,
               ymin = 0, ymax = max(top_items$total_unidades), color = "red",
               fill = NA, size = 1.2)

    ItemsApronosticar <- top_items[c(1,2,14,15,29,30),1]


    # Buscar la carpeta con el nombre de "Tienda" dentro de "TransformacionDeDatos"
    carpeta_existente <-
      drive_get(path = file.path("Proyecto R", "TransformacionDeDatos", as.character(Tienda)))
    if (nrow(carpeta_existente) > 0) {
      drive_rm(carpeta_existente)
    }

    carpeta_proyecto <- drive_get(paste0("Proyecto R/TransformacionDeDatos"))
    carpeta_tienda <- drive_mkdir(as.character(Tienda), path = carpeta_proyecto)

    # Crear un archivo temporal en tu sistema local
    archivo_temp <- tempfile(fileext = ".csv")
```

```
    write.csv(ItemsApronosticar, archivo_temp, row.names = FALSE)
    # Subir el archivo a la carpeta de la tienda en Google Drive
    drive_upload(media = archivo_temp, path = carpeta_tienda, name = paste0(Tienda,".csv"))
    #Eliminar el archivo temporal si ya no lo necesitas
    unlink(archivo_temp)

    return(p)

}
```

Explicacion por segmento de codigo:

```
    primer_dia_hace_dos_meses <- floor_date(max(UnidadesVendidas$fecha) %m-% months(1), "month")

    DatosTienda <- UnidadesVendidas[UnidadesVendidas$centro_op == Tienda &
                        UnidadesVendidas$fecha >= primer_dia_hace_dos_meses,]
```

1. Cálculo de la fecha de inicio: En la variable primer_dia_hace_dos_meses, calculamos el primer día del mes que corresponde a dos meses antes de la fecha máxima en UnidadesVendidas. Por ejemplo, si la última fecha es "2024-06-30", primer_dia_hace_dos_meses será "2024-05-01", lo que nos permite filtrar las ventas de los últimos dos meses.

La variable DatosTienda recibe el identificador de una tienda y utiliza primer_dia_hace_dos_meses para filtrar UnidadesVendidas, generando un dataframe con las ventas de los últimos dos meses para esa tienda.

```
    top_items <- DatosTienda %>%
      group_by(item) %>%
      summarize(total_unidades = sum(unidades)) %>%
      arrange(desc(total_unidades))  %>%
      head(30)
```

2. Filtrado de datos por tienda y fecha: Una vez filtrados los datos de unidades vendidas por tienda para los últimos dos meses, se agrupan por producto (item) y se suman las unidades vendidas de cada uno. Luego, se ordenan en orden descendente según la cantidad total de unidades vendidas, seleccionando los 30 productos más vendidos utilizando head(30).

```
    p <- ggplot(top_items, aes(x = reorder(item, -total_unidades), y = total_unidades,
                               fill = total_unidades)) +
      geom_bar(stat = "identity") +
      scale_fill_gradient(low = "lightblue", high = "steelblue") +
      scale_y_continuous(labels = comma) +
      labs(title = paste0("Top 30 Items Más Vendidos, Tienda: ",Tienda), x = "Item",
           y = "Total Unidades Vendidas") +
      theme_minimal() +
      theme(
        legend.position = "none",
        axis.text.x = element_text(angle = 45, hjust = 1),
        axis.text.y = element_blank()
      ) +
      annotate("rect", xmin = 0.5, xmax = 2.5,
               ymin = 0, ymax = max(top_items$total_unidades), color = "red",
               fill = NA, size = 1.2) +
```

```
    annotate("rect", xmin = 13.5, xmax = 15.5,
             ymin = 0, ymax = max(top_items$total_unidades), color = "red",
             fill = NA, size = 1.2) +
    annotate("rect", xmin = 28.5, xmax = 30.5,
             ymin = 0, ymax = max(top_items$total_unidades), color = "red",
             fill = NA, size = 1.2)
```

3. Creación de un gráfico: En el anterior segmento de codigo lo que hacemos es crear un gráfico de barras con los 30 productos más vendidos utilizando ggplot2, apartir de la data que ya viene filtrada del paso anterior. Además, se destacan con rectángulos rojos las posiciones de los productos seleccionados los dos más vendidos (1,2), los dos de ventas medianas (14,15), y los dos menos vendidos (29,30).

```
ItemsApronosticar <- top_items[c(1,2,14,15,29,30),1]
```

4. Selección de los productos para el pronóstico: ItemsApronosticar guarda las posiciones de los datos filtrados para posteriormente cargar un archivo CSV en nuestro drive, como se explica a continuación.

```
# Buscar la carpeta con el nombre de "Tienda" dentro de "TransformacionDeDatos"
carpeta_existente <-
  drive_get(path = file.path("Proyecto R", "TransformacionDeDatos", as.character(Tienda)))

if (nrow(carpeta_existente) > 0) {
  drive_rm(carpeta_existente)
}

carpeta_proyecto <- drive_get(paste0("Proyecto R/TransformacionDeDatos"))
carpeta_tienda <- drive_mkdir(as.character(Tienda), path = carpeta_proyecto)

# Crear un archivo temporal en tu sistema local
archivo_temp <- tempfile(fileext = ".csv")
write.csv2(ItemsApronosticar, archivo_temp, row.names = FALSE)

# Subir el archivo a la carpeta de la tienda en Google Drive
drive_upload(media = archivo_temp, path = carpeta_tienda, name = paste0(Tienda,".csv"))

#Eliminar el archivo temporal si ya no lo necesitas
unlink(archivo_temp)
```

5. Gestión de archivos en Google Drive: carpeta_existente guarda el identificador único creado por Google Drive para cada carpeta. Cada carpeta lleva el mismo nombre que la tienda ingresada en la función. En la línea del if, se valida si la carpeta ya existe; si el identificador para esa carpeta existe, se procede a eliminar dicha carpeta.

La variable carpeta_proyecto almacena la ruta donde se guardarán los productos a pronosticar. carpeta_tienda crea una nueva carpeta con el nombre de la tienda en la ruta especificada por carpeta_proyecto.

archivo_temp crea un archivo temporal en memoria, y write.csv escribe el contenido de ItemsApronosticar en este archivo temporal. Luego, drive_upload sube el archivo temporal a la nueva carpeta creada en Google Drive, utilizando el nombre de la tienda para el archivo CSV. Finalmente, unlink(archivo_temp) elimina el archivo temporal de la memoria.

```
return(p)
```

6. Mostramos el grafico: Mostramos el grafico almacenado en el paso **3.Creación de un gráfico**.

Ahora ejecutemos la función para cada una de nuestras tiendas elegidas en nuestra clusterizacion y validemos los resultados obtenidos, con la creacion de la carpeta en google drive en la ruta **Proyecto R/TransformacionDeDatos** con los identificadores de los productos a pronosticar.

```
Funct_TopItems(102)
```

```
> Funct_TopItems(102)
✓ The input `path` resolved to exactly 1 file.
File deleted:
• 102 <id: 1APeBdZIDF2sqLGREb8AArTAhcCgMuvzi>
✓ The input `path` resolved to exactly 1 file.
Created Drive file:
• 102 <id: 1o6A7ELdM0fz03p        DSuVLmnKnw>
With MIME type:
• application/vnd.google-apps.folder
Local file:
• C:\                    .AppData\Local\Temp\RtmpWUnzzm\file915035b26551.csv
Uploaded into Drive file:
• 102.csv <id: 15Al3UhDp6S8DoGV.       7bmf8QTho3>
With MIME type:
• text/csv
```

Figura 4: Ejecucion de la funcion `Funct_TopItems(102)` con tienda.

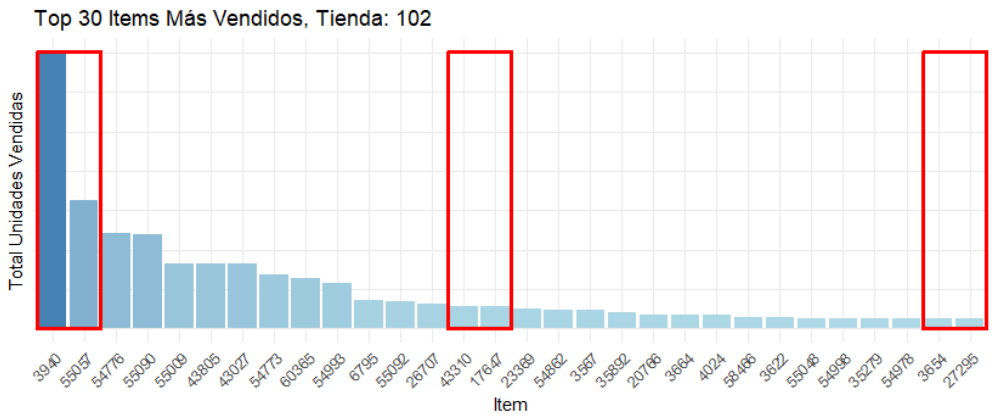


Figura 5: Grafico resultado de la funcion `Funct_TopItems(102)`.

Con los pasos previos para la muestra intencionada basada en la clasificación de unidades vendidas, aseguramos la selección idónea de una muestra de productos a pronosticar, considerando diferentes clasificaciones y comportamientos en las series de tiempo de las tiendas. A continuación, emplearemos el primer modelo para realizar los pronósticos de las tiendas seleccionadas, con sus respectivos productos almacenados en la carpeta de Google Drive Proyecto R/TransformaciónDeDatos. Cada subcarpeta dentro de esta ruta tiene el mismo identificador que la tienda correspondiente a pronosticar.