Trabajo Fin de Máster

Problemas de optimización: Machine Learning vs Métodos tradicionales

Mauro Vidal Antepazo

Máster en Técnicas Estadísticas

Curso 2024-2025

Propuesta de Trabajo Fin de Máster

Título en galego: Problemas de optimización: Machine Learning vs Métodos tradicionais

Título en español: Problemas de optimización: Machine Learning vs Métodos tradicionales

English title: Optimisation problems: Machine Learning vs. traditional methods

Modalidad: Modalidad B

Autor/a: Mauro Vidal Antepazo, Universidad de Santiago de Compostela

Director/a: Beatriz Pateriro Lopez, Universidad Santiago de Compostela; Julio González Díaz, Universidad Santiago de Compostela

Tutor/a: Javier Soria Esponera, SDG Group; Daniel Tarazon Barbera, SDG Group

Breve resumen del trabajo:

Desde la consultora SDG se presentó como objetivo de este proyecto la realización de una comparación en términos de eficiencia económica entre dos enfoques distintos a la hora de definir los valores de un modelo MIN-MAX. Todo proceso de gestión de inventario necesita un proceso de predicción de demanda y un proceso de optimización de recursos. El objetivo del presente proyecto es dirimir si el enfoque clásico de predicción de la demanda y optimización posterior es sustancialmente mejor que un nuevo enfoque, que realiza un proceso de optimización de los valores MIN-MAX en el histórico de datos y utiliza estos valores para realizar la predicción.

Recomendaciones:

Otras observaciones:

Declaración responsable. Para dar cumplimiento a la Ley 3/2022, de 24 de febrero, de convivencia universitaria, referente al plagio en el Trabajo Fin de Máster (Artículo 11, Disposición 2978 del BOE núm. 48 de 2022), **el/la autor/a declara** que el Trabajo Fin de Máster presentado es un documento original en el que se han tenido en cuenta las siguientes consideraciones relativas al uso de material de apoyo desarrollado por otros/as autores/as:

- Todas las fuentes usadas para la elaboración de este trabajo han sido citadas convenientemente (libros, artículos, apuntes de profesorado, páginas web, programas,...)
- Cualquier contenido copiado o traducido textualmente se ha puesto entre comillas, citando su procedencia.
- Se ha hecho constar explícitamente cuando un capítulo, sección, demostración,... sea una adaptación casi literal de alguna fuente existente.

Y, acepta que, si se demostrara lo contrario, se le apliquen las medidas disciplinarias que correspondan.

Índice

Resumen

Resumen en español

Desde la consultora SDG se presentó como objetivo de este proyecto la realización de una comparación en términos de eficiencia económica entre dos enfoques distintos a la hora de definir los valores de un modelo MIN-MAX. Todo proceso de gestión de inventario necesita un proceso de predicción de demanda y un proceso de optimización de recursos. El objetivo del presente proyecto es dirimir si el enfoque clásico de predicción de la demanda y optimización posterior es sustancialmente mejor que un nuevo enfoque, que realiza un proceso de optimización de los valores MIN-MAX en el histórico de datos y utiliza estos valores para realizar la predicción.

English abstract

From the SDG consultancy, the objective of this project was presented as a comparison in terms of economic efficiency between two different approaches for defining the values of a MIN-MAX model. Every inventory management process requires both a demand forecasting process and a resource optimization process. The objective of this project is to determine whether the classical approach—forecasting demand and then optimizing—is substantially better than a new approach, which performs an optimization of the MIN-MAX values on historical data and uses these values to make the forecast.

Capítulo 1

Introducción problema

El presente trabajo se clasifica en la modalidad de TFM B. Esto implica que la propuesta del trabajo está definida por una empresa, en este caso, la consultora SDG Group. El objetivo de la empresa con este trabajo ha sido doble. Por un lado, tenían interés en desarrollar una prueba de concepto en torno a los problemas de gestión de inventario en sectores minoritarios (el foco principal fue consumo minoritario de alimentación). Por otro lado, aprovechando que la implementación de un sistema de gestión de inventario consta de dos partes diferenciadas, optimizar y predecir, se planteó como objetivo la comparación de dos enfoques distintos a la hora de abordar este problema, que se basa en aplicar los pasos de optimización y predicción en distinto orden.

1.1. Introducción a la gestión de inventarios

El término inventario está fuertemente ligado con procesos industriales o empresariales, pero aquí queremos introducirlo de una forma más abstracta. Que nos permita entender este concepto y cómo tiene que estar ligado inequívocamente de la palabra gestión (o previsión). Pongamos un ejemplo desligado de un proceso comercial (donde existe un intercambio de bienes), pensemos en una persona que tiene un cultivo a partir del cual intentará abastecer todo su consumo anual. Está claro que nos encontramos con un problema. Es imposible sincronizar el tiempo de producción con el tiempo de consumo, por ello, como consecuencia 'natural' nace un excedente de producción en un determinado tiempo. Este excedente se almacenará con el objetivo de satisfacer demandas futuras, esto es lo que conoceremos como inventario. La determinación de la cantidad de producción necesaria requiere de una previsión del consumo futuro (habitualmente delimitado en una franja temporal).

A nivel histórico, la llegada de la revolución industrial y la extensión de la capacidad de comercialización han sido el germen de sociedades de trabajo especializado. Estos procesos históricos han dado lugar a industrias que producen grandes cantidades de bienes específicos que abastecen a una gran cantidad de consumidores. Como los procesos productivos, en analogía con el ejemplo expuesto, requieren de tiempos distintos al de venta/consumo, la gestión de inventarios ha tomado cada vez un mayor peso en la eficiencia económica de las empresas. Una de las herramientas claves en la mejora de la gestión de inventarios ha sido la introducción de conceptos matemáticos y técnicas que han facilitado los procesos de predicción.

La primera aproximación metodológica a la gestión de inventarios se presenta en 1913 con el artículo *How Many Parts to Make at Once* [1] de Whitman Harris, donde se presenta el modelo EOQ (Economic Order Quantity). Podemos ver en la figura 1.1 el resultado

gráfico de la evolución del inventario bajo este modelo de gestión.

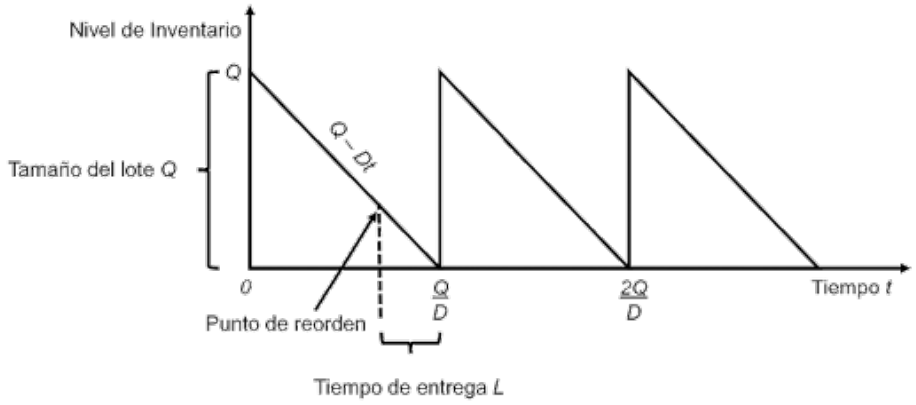

Figura 1.1: Gráfica evolución stock bajo modelo EOQ. Referencia fotografía: http://virtual.umng.edu.co/distancia/ecosistema/ovas/diplomados/diplomado_de_logistica_integral/unidad_3/medios/documentacion/img/p4h1f1.png Enlace

Este artículo fue pionero en el estudio sistemático de los modelos de gestión de inventario. Uno de los aportes fundamentales de dicho estudio es la estructuración de un conjunto de conceptos y suposiciones necesarias para determinar reglas claras que nos permitan gestionar los inventarios de forma mecanizada.

El primer paso es caracterizar el término de modelo de inventaria. Nos decantamos por una definición sencilla: diremos que un **modelo de inventario** es cualquier procedimiento que determina cuándo realizar un reabastecimiento (el cual puede estar motivado por diversas acciones, como la compra o la fabricación de producto) y cuál debe ser su tamaño.

Estos modelos inevitablemente necesitan realizar ciertas suposiciones previas sobre las que trabajar. La modelización de la demanda será una de las suposiciones claves a la hora de definir el modelo. Si caracterizamos la demanda como un parámetro conocido, diremos que el modelo es determinístico. La otra opción es definir la demanda como una variable aleatoria, lo que da lugar a un modelo estocástico. En nuestro caso nos centraremos únicamente en el caso de una demanda conocida, que será estimada mediante técnicas de predicción de series temporales.

Además del supuesto sobre la demanda, necesitamos añadir otros que sirvan de sustento de la modelización de un proceso real. Por eso, también necesitaremos conocer el tiempo de entrega, que será el tiempo transcurrido entre que se lanza la orden de reabastecimiento y la llegada del material.

Otro de los puntos claves a definir en los modelos de gestión de inventarios viene dado por la restricción de abastecimiento completo de la demanda. Esta restricción implica que la solución que aporta nuestro modelo necesita estrictamente satisfacer toda la demanda que se haya determinado. Por ejemplo, en el modelo EOQ esta es una restricción que se incluye en el modelo. Sin embargo, se puede relajar esta restricción en otros modelos, permitiendo *Rupturas de la demanda* que llevarán un coste asociado. Bastaría con incluir en el coste, el beneficio no realizado cuando ocurre la pérdida de una venta de un producto determinado.

En lo que se refiere al supuesto anterior, es necesario también introducir un coste a minimizar en la gestión. Este estará determinado con el conjunto de presupuestos que hayamos definido en nuestro problema. Por ejemplo, en un caso en el que se permite al modelo tener rupturas de la demanda, necesitaremos conocer el beneficio asociado a cada

venta, para imputar el beneficio no realizado como una pérdida asociada a la gestión. Otros ejemplos pueden ser costes asociados a la realización de pedidos, coste de tirado de productos caducados,...

Los conceptos presentados serán los elementos principales de los modelos que se irán presentando a lo largo del texto. Estos no son únicos, y las particularidades de negocio requerirán añadir estas características mediante otras suposiciones.

La selección del modelo ha sido prefijada por SDG de forma apriorística. El modelo seleccionado de antemano es conocido como MIN-MAX, que consiste en la definición de un valor mínimo de inventario, un umbral que permite realizar la decisión de reabastecimiento cuando nuestra cantidad de inventario sobrepase dicha marca. La otra variable de decisión del modelo será el valor MAX, que determina el valor de inventario que se debe de alcanzar en cada reabastecimiento (lo que determinará una cantidad de reabastecimiento distinta en cada caso). Las gráficas que representan la variación temporal de un inventario presentan un formato de dientes de sierra. En nuestro modelo, esta tendencia vendrá determinada por los valores de MIN-MAX que están vigentes en ese momento. Ponemos un ejemplo gráfico de este fenómeno (ver figura 1.2).

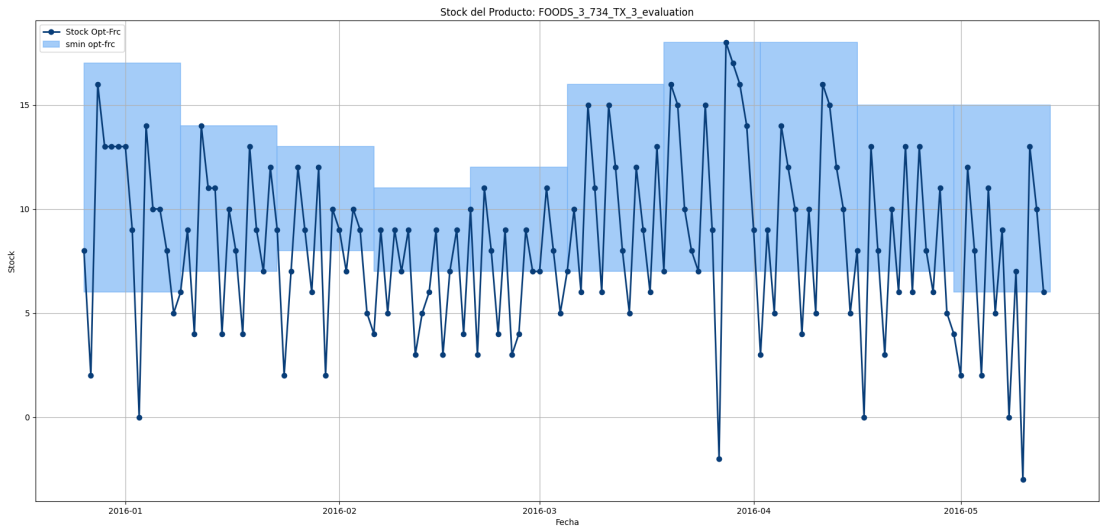

Figura 1.2: Gráfica evolución de inventario bajo un modelo MIN-MAX (con valores MIN-MAX variante cada 14 días)

1.2. Series temporales

La estructura teórica que se utiliza en la predicción de la demanda para modelizar el histórico de ventas son las series temporales. Es por ello que haremos una breve introducción teórica de este término, buscando dar consistencia a la terminología usada en todo el trabajo.

Definición 1.1. Un **proceso estocástico** es un conjunto de variables aleatorias,$\{X_t\}_{t\in C}$ definidas sobre el mismo espacio de probabilidad.

En este caso, el conjunto C será el de los números enteros $\mathbb{Z}$, por lo tanto, el proceso estocástico tendrá la siguiente formulación:

$$..., X_{-2}, X_{-1}, X_0, X_1, X_2, ... \tag{1.1}$$

Cada elemento de la expresión anterior hace referencia a una variable aleatoria en el instante t.

Definición 1.2. Definimos como **realización o trayectoria** a las observaciones de un proceso estocástico.

Denotaremos la realización o trayectoria con la siguiente notación:

$$..., x_{-2}, x_{-1}, x_0, x_1, x_2, ... \tag{1.2}$$

Definición 1.3. Se define como **serie temporal** a una realización o trayectoria parcial de un proceso estocástico.

$$x_1, x_2, ..., x_T \tag{1.3}$$

Pongamos algunos ejemplos de procesos estocásticos que generan una serie temporal:

- Ruido blanco es una colección de variables aleatorias incorreladas, con media 0 y varianza finita σ_a^2. Denotamos por $\{a_t\}_t$.
- MA(1) (Media móvil de orden 1) determinada por la ecuación $X_t = c + a_t + \theta_1 \cdot a_{t-1}$, donde $\{a_t\}_t$ es ruido blanco, y c y θ_1 ($\theta_1 \neq 0$) son parámetros.
- Paseo aleatorio, dada por la ecuación $X_t = c + X_{t-1} + a_t$ donde $\{a_t\}$ es ruido blanco.

Podemos observar gráficamente, en la Figura 1.3, las series temporales generadas a partir de los procesos estocásticos que hemos puesto como ejemplo.

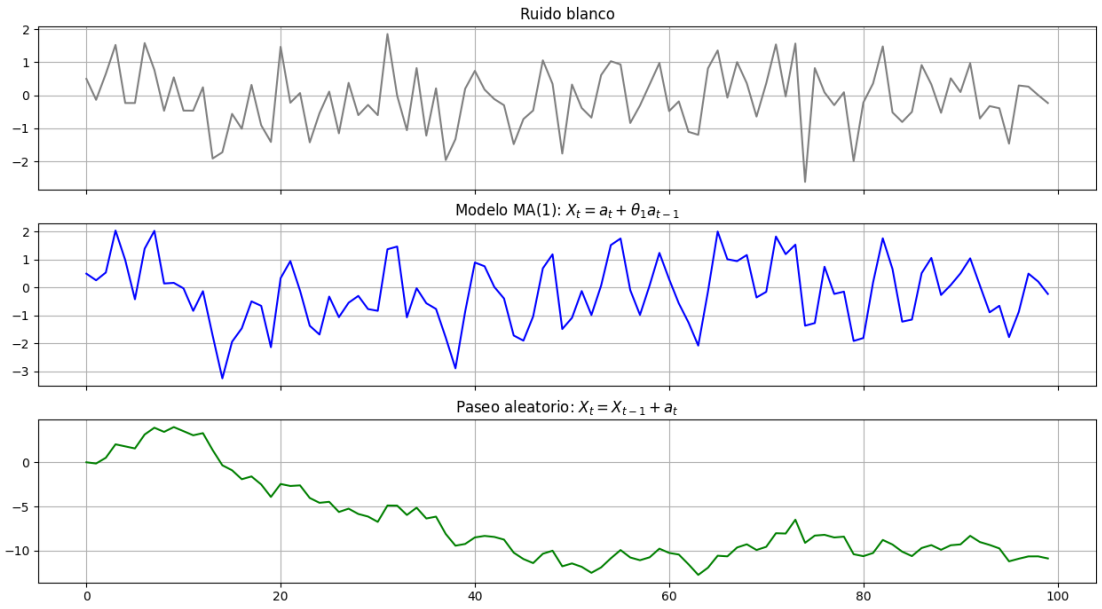

Figura 1.3: Ejemplos procesos estocásticos

En el trabajo, las series temporales serán utilizadas para modelizar el histórico de ventas de cada producto de nuestra tienda. Un enfoque clásico del proceso de predicción consistiría en estimar un proceso estocástico que genere los datos que tenemos en el histórico de ventas. En nuestro caso, con el objetivo de desarrollar una solución que sea utilizable por tiendas con un gran catálogo de productos, nos hemos decantado por buscar enfoques y técnicas globales. Entendemos como soluciones globales aquellos modelos de predicción

que utilizan datos de diferentes series temporales en su entrenamiento (utilizando una penalización agregada de las diferentes series).

En este caso, la propuesta por parte de los expertos de la empresa fue hacer uso de modelos de Aprendizaje Profundo que están diseñados para utilizar grandes cantidades de datos en su proceso de entrenamiento.

1.3. Presentación del problema

1.3.1. Conocimiento negocio y objetivos proyecto

Como ya hemos expuesto al inicio, nuestro proyecto se engloba dentro de las pruebas de concepto o experimento de SDG, es por eso que no contamos con un conocimiento de negocio tan completo como podríamos haber tenido de haber realizado este proyecto de la mano de una gran firma del sector minorista. Por eso hemos tenido que utilizar un dataset público que contuviera toda la información necesaria para llevar el proyecto a cabo. Por otro lado, a la hora de establecer los requerimientos de negocio sobre la solución, nos hemos decantado por trabajar con los supuestos más generales posibles, creando así una solución fácilmente adaptable a problemas reales.

El objetivo inicial es claro: desarrollar una herramienta que permita a los supermercados realizar una gestión eficiente del inventario. La restricción del tipo de negocio es importante a la hora de modelizar nuestro problema, ya que, este modelo de negocio está caracterizado por la demanda intermitente, los productos perecederos, limitación de capacidad de almacenaje, gestión de proveedores, entre otras cosas. Todas estas características hacen este modelo de negocio más susceptible a problemas con la gestión de inventario. Además, añaden complejidad a la hora de definir los supuestos necesarios en la modelización del problema. Esta alta complejidad es la que puede justificar económicamente el uso de técnicas estadísticas avanzadas para la determinación de las reglas de reabastecimiento.

La experiencia de SDG en proyectos reales similares es la que los llevó a decantarse por trabajar con un modelo MIN-MAX. Tienen la certeza de que grandes compañías dentro del sector hacen uso de este modelo como herramienta en la gestión del inventario. Uno de los puntos destacados desde la compañía es que los clientes con los que han trabajo en proyectos similares hacen uso de ese modelo, pero manteniendo los valores de MIN-MAX inalterados por largos períodos de tiempo (+6 meses). Por ello, se ha fijado como objetivo de minería del proceso la determinación de los valores MIN-MAX a nivel de producto con una validez de 14 días. Este enfoque busca que la herramienta sea adaptativa y que reduzca, los costes asociados a la gestión de inventario. También, en los casos en los que este proceso no esté automatizado, la creación de un proceso automático de gestión de inventario puede reducir costes operativos en las empresas que lo implementen.

Después de un proceso de discusión interna, se acordó fijar las siguientes características o criterios de negocio que van a determinar el proceso de modelización matemática del problema de optimización asociado.

- La tienda contará con una capacidad máxima determinada.
- Conocemos el histórico de datos de demanda de la tienda.
- Existen productos perecederos.
- La tienda depende de varios proveedores para la gestión de su stock.
- Las recargas de stock, implicarán el envío de mercancía por parte de los proveedores, los cuales tendrán un coste asociado.

- Conocemos el coste de todos los productos.
- Conocemos el precio de venta al consumidor.
- Se permite la ruptura de stock dentro del modelo.
- Se implementa un criterio de ICM (Imagen Comercial a Mantener), lo que se traduce en un coste asociado cuando el inventario de un producto baja de ese umbral.
- Sistema FIFO (First IN Firts Out).

Uno de los problemas que acarrea no realizar este proyecto en un cliente real, es que no conocemos un modelo base (actual) donde podamos medir las mejoras que nuestros modelos puedan aportar en términos de eficiencia. Este ha sido otro de los puntos que ha necesitado de la realización de suposiciones por nuestra parte. Como consideramos clave tener un modelo base donde comparar nuestros resultados, hemos intentado ajustar un criterio de reabastecimiento sencillo que fuera compatible con el conocimiento que desde SDG se tiene de soluciones actuales en el sector. Este modelo base nos servirá como criterio para evaluar el rendimiento de las soluciones que vayamos obteniendo (y que usamos como comparación en el tercer capítulo).

El modelo base está justificado mediante 2 criterios de negocio claros:

- Definir el **MIN** como el valor del día previo al que se que tendré una ruptura de inventario. El conocer el inventario al inicio del periodo y todas las demandas de la ventana temporal hace que pueda calcular el día en que me quedaré sin inventario, para intentar prevenir esa ruptura, definimos el valor MIN como el inventario que tendremos el día previo (y así se lanzará la orden de reabastecimiento).
- Para definir el **MAX**, contaremos también con el conocimiento del valor del MIN (siguiendo el criterio anterior). En este caso, hemos definido como regla estándar que el tamaño de nuestro reabastecimiento será igual al acumulado de las demandas de los 3 días posteriores al día que determina el valor del MIN. Este criterio hace que el valor de MAX sea el valor de MIN más la demanda agregada de los 3 días posteriores a la fecha en la que tomamos el inventario como valor de MIN.

Veamos el siguiente ejemplo. Partimos de un valor de 25 unidades de inventario de un producto de nuestra tienda y tenemos la siguiente serie de demandas $[6, 2, 0, 7, 3, 3, 1, 6, 1, 3, 3, 2, 2]$. Esto hace que, si nos ponemos en el supuesto de no reabastecimiento, en el séptimo día tenemos 3 unidades en inventario, pero la demanda del día siguiente es de 7, lo que provocaría una ruptura de inventario. Siguiendo el criterio expuesto, fijamos el valor de MIN a 3 (que es el inventario del día previo a no satisfacer la demanda), ver punto rojo en la Figura 1.4.

El siguiente paso es la determinación del valor de MAX. Conocemos que en el séptimo día es el día previo a la ruptura de inventario y que por eso hemos definido el MIN=3. Ahora, como criterio de nuestro modelo base, realizamos un reabastecimiento del tamaño del agregado de demanda de los próximos 3 días, que son los días 8,9,10 que representan las siguientes demandas (6,2,3) (representamos en verde la acumulación de esos días 1.4). Siguiendo este criterio, la variable MAX toma el valor MAX=3 (MIN)+11 (Acumulado de demanda)=14.

El modelo base es adaptativo, se define también cada 14 días. La experiencia interna de SDG en proyectos similares es que los procesos actuales del sector hacen uso de soluciones que perduran mucho más en el tiempo. Es por esto que consideramos que nuestra solución base supone un reto importante, ya que podría estar superando a métodos de decisión muy rígidos y que se hacen de forma manual.

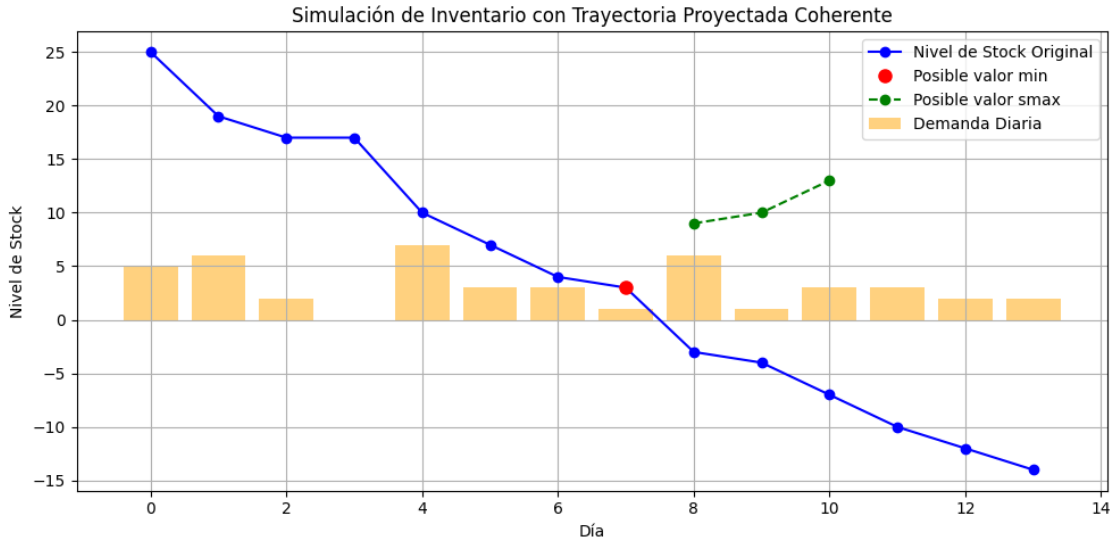

Figura 1.4: Modelo base

1.3.2. Datos

Los datos utilizados en el estudio provienen de un dataset público muy conocido. En 2020 se organizó un concurso internacional de predicción de series temporales organizado por Spyros Makridakis donde se facilitó (de forma pública) un dataset que contenía datos de ventas del supermercado más grande de EE. UU., Walmart. El dataset contiene una basta información de ventas de 40000 productos, además de incluir datos adicionales como: precios, promociones, calendario... Los resultados del concurso fueron publicados en un artículo encabezado por el organizador [2].

La información expuesta en dicho concurso estaba estructurada en 3 datasets, que serán las fuentes de datos de nuestro trabajo:

- **sales_train_evaluation**: Este dataset contiene el histórico de ventas de más de 40000 productos junto con algunas variables de contexto (departamento, estado y tienda) que determinan una forma estructura del dataset.
- **sell_prices**: Este dataset contiene los datos referentes al precio semanal de cada producto.
- **calendar**: Este dataset contiene información relacionada con las fechas que permite contextualizar las ventas diarias. Recoge variables de contexto relevantes como eventos especiales (por ejemplo, NBA Finals, Navidad, etc.) y la presencia de subsidios alimentarios (SNAP) en distintos estados, lo cual puede influir en el comportamiento de compra.

De nuevo, la gestión de un proyecto interno ha impedido que contemos con toda la información necesaria para satisfacer todos los presupuestos que hemos presentado hasta el momento. Es por esto que ciertos datos y parámetros han sido generados por nuestra cuenta. El foco ha sido siempre mantenernos en un nivel mínimo de información generada. En la generación de información se ha puesto énfasis en diseñar procesos que fueran lo más realistas posible.

A continuación, presentamos los parámetros que han sido generados por nuestra cuenta y el proceso que utilizamos en la generación de cada uno de ellos:

- Stock inicial: Se generó un valor aleatorio entre 2.5 y 3.5, que sirvió como multiplicador del percentil 90 del histórico de la demanda.
- Capacidad máxima: Aleatoria entre 2.5 y 3 veces el día de más demanda del histórico.
- Coste producto: Para el cálculo del coste, se le asigna a cada producto un porcentaje entre el 70 % y 90 % del precio de venta conocido.
- Coste de envío: El coste del envío se genera de forma aleatoria entre 20 €y 100 €.
- ICM: Con el objetivo de que la suma del IMC de todos los productos fuera el 5 % de la capacidad máxima, se realizó una asignación aleatoria que consistió en 2 etapas. Una primera etapa en la que al 50 % de los productos se le asigna un ICM de 0. Después, en una segunda etapa, al resto de productos se le asigna un porcentaje respecto de los productos del 'IMC global' (5 % de la capacidad máxima de la tienda).
- Vida Útil: En este punto se buscó limitar la posibilidad de que se asignaran vidas útiles muy similares a productos con ventas altas o costes altos. Para ello, se planteó una agrupación de los productos basada en una estimación no paramétrica de la densidad sobre un espacio bidimensional determinado por las variables: Nº de ventas, precio. Se crean 10 grupos y se genera un vector de valores para la vida útil (entre 1 y 30) de tamaño 100, que se asigna de forma aleatorio a cada grupo de productos. Este proceso permite tener en todos los grupos vidas útiles variadas.

1.3.3. Presentación de enfoques

La presentación hasta el momento menciona el objetivo global que SDG tiene respecto al desarrollo de este proyecto. Sin embargo, otra de las patas claves del trabajo consiste en comparar 2 enfoques de resolución de la problemática planteada. Estos enfoques comparten componentes comunes; ambos constan de un proceso de optimización y otro de predicción, pero la concatenación de estas técnicas se aplica de forma inversa. Esto hace que el proceso de minería sea distinto, aunque partamos de un mismo conjunto de datos y nuestro output final tenga también el mismo formato (los valores MIN-MAX).

Enfoque Predicción-Optimización

Este primer enfoque, se podría denominar como el enfoque clásico. Partiendo de los datos históricos de ventas, entraríamos en una primera etapa en la que utilizamos técnicas de predicción de series temporales (en este caso modelos de Aprendizaje Profundo) para obtener una predicción de la demanda futura. La segunda de las etapas hará uso de los datos de demanda predichos para, mediante técnicas de optimización, buscar los valores de MIN-MAX que sean más rentables para el negocio.

Enfoque Optimización-Predicción

El segundo enfoque nace de la idea de revertir el orden habitual del proceso anterior. En este caso, el primer proceso es el de optimización. En primer lugar, se utilizan métodos de optimización para calcular en ventanas de 14 días, los valores óptimos de MIN-MAX que debería haber usado negocio y se construye así un historial de valores de MIN-MAX por producto. El segundo proceso se plantea con el objetivo de predicción, donde se construyen las series temporales con los valores de MIN-MAX calculados con anterioridad.

Uno de los problemas a la hora de realizar la comparación entre enfoques es que el enfoque Optimización-Predicción requiere de una cantidad de cómputo muy elevada en un inicio. Teniendo en cuenta que el objetivo de minería se ha fijado en ventanas de 14 días, el tener un histórico de más de 5 años hace que la puesta en marcha de este enfoque (primer paso de optimización) necesite de ejecutar el cálculo de un problema de optimización en cada ventana, siendo un total de 130 ejecuciones. Esto ha significado un reto en el desarrollo del proyecto y nos ha llevado a tomar la decisión de limitar la ejecución máxima del proceso de optimización a 1 hora. Esto se traduce en que el cálculo del histórico de MIN-MAX tardaría casi 5 días y medio (de ejecución ininterrumpida).

Aun así, esta restricción también se ha aplicado al otro enfoque, ya que consideramos que una comparación justa debe ser realizada permitiendo el mismo tiempo de cómputo a las dos soluciones (aunque el Optimización-Predicción necesite un primer cálculo único muy costoso).

Capítulo 2

Técnicas utilizadas

Los dos enfoques presentados requieren de la realización de 2 tareas comunes: Optimizar y Predecir. Aunque la aplicación de estas técnicas requiere cierta adaptación en cada uno de los enfoques, debido a la entrada y salida de datos que se espera en cada caso, intentamos presentar las técnicas utilizadas de forma conjunta (caracterizando la personalización a los enfoques cuando sea necesario).

2.1. Optimización

De forma teórica, un problema de optimización matemática consiste en la búsqueda de un valor de una variable x en un conjunto de decisión X determinado por una serie de restricciones donde se busca minimizar o maximizar una función objetivo $f(x)$. En nuestro caso, el proceso de optimización tratará de devolver los valores de MIN-MAX óptimos a nivel de producto para los periodos bisemanales fijados. Este proceso ha sido tedioso y caracterizado por una búsqueda asistemática. Se han realizado numerosas pruebas con diferentes bocetos (que no incluían la formulación objetiva planteada). En este apartado, introducimos de forma breve todas estas pruebas que se han realizado antes de llegar a la solución seleccionada, basada en un algoritmo genético. Cabe destacar que el hilo conductor en la búsqueda de un proceso de optimización adecuado ha sido la mejora de la solución base planteada, limitado por la duración máxima de 1h de cómputo.

2.1.1. Modelización matemática

El primer objetivo fue modelizar el problema de optimización matemática asociado al problema de negocio presentado y hacer uso de solvers para la obtención directa de un óptimo global. El proceso fue iterativo, desde un problema sencillo (que prescindían de algunos supuestos) hasta modelizaciones que recogían mayor complejidad del proyecto objetivo. Por simplificar la exposición de esta parte, hemos decidido centrarnos únicamente en la modelización más completa que hemos llegado a implementar. Aun así, cabe destacar que esta implementación no tiene en cuenta los costes que produce el tirado de productos caducados, lo cual seguía siendo una gran simplificación. Con este planteamiento se busca trasladar que incluso con la modelización de un problema más sencillo al objetivo, no se han conseguido resultados aceptables que combinen optimalidad de la solución y eficiencia de cómputo.

Recordemos que basamos la gestión de inventario de nuestra tienda en dos variables que van a controlar el mínimo y el máximo de inventario que nos interesa tener por producto.

Además, en consonancia con los criterios logísticos y de negocio presentados en el primer capítulo, tendremos que tener en cuenta los siguientes supuestos:

- La tienda tiene una capacidad máxima. Esto determinará la cantidad máxima de productos que podemos acumular de forma conjunta.
- Nuestro catálogo de productos está distribuido entre un conjunto de proveedores. Cada producto nos lo distribuye un único proveedor, pero un proveedor puede suministrarnos varios productos.
- El criterio de control se basa en 2 reglas: si el nivel de inventario de un producto baja del umbral determinado por el valor mínimo, realizamos un pedido de ese producto. El pedido tiene que traer la cantidad necesaria de producto para llenar el inventario hasta el valor del umbral máximo.
- Todos los pedidos tienen un coste base (independiente de la cantidad de productos) que es distinto en cada proveedor.
- La tienda tiene una capacidad logística limitada, no podremos realizar pedidos infinitos para un mismo día. El número de pedidos gestionados por día estará limitado.

Uno de los pasos en la construcción de un problema de optimización matemática es definir la función objetivo (que en este ejemplo de modelización deja fuera el coste de tirado). Tenemos que definir una función que recoja los diferentes gastos derivados de la gestión de inventario de la tienda. Para ello, troceamos la función en diferentes costes que explicamos a continuación:

- Coste 1: Si sobrepasamos la capacidad máxima de la tienda, los productos sobrantes tendrán que ser tirados (suponiendo el coste asociado la media del coste de compra de los productos).
- Coste 2: Coste estándar de envío por proveedor.
- Coste 3: Coste asociado a la ruptura de stock, es decir, cuando dejamos demanda sin abastecer. Como conocemos el precio de compra de los productos y el precio de venta, imputaremos un coste de ruptura de stock determinado por la diferencia entre el precio de venta esperado y el precio de compra.

El siguiente paso es determinar las diferentes variables y parámetros que vamos a utilizar para expresar estos requerimientos de negocio en lenguaje matemático:

- **Conjuntos:**
  - Productos:= $p \in \{1, ..., P\}$.
  - Días:= $t \in \{1, ..., T\}$.
  - Proveedores:= $s \in \{1, ...S\}$.
- **Variables:**
  - $stock[p, t]$: medición del inventario de un producto al final de un día.
  - $stock_{rup}[p, t]$: variable de holgura que determina la demanda sin abastecer de un producto en un día determinado.

- $cap_{hol}[t]$: variable de holgura para la capacidad máxima de la tienda que se encarga de medir la cantidad de productos que ha sobrepasado la capacidad de la tienda para cada día.
- $smin[p], smax[p]$: variables de control de decisión por producto que modelizan el criterio basado en 2 reglas. Realizamos el pedido si $stock < smin$ y pedimos lo necesario para que $stock = smax$ (siendo este un stock teórico, al principio del día, la relación real con la variable $stock$ sería la siguiente: $stock[p,t] = smax[p] - DEM[p,t]$) y por tan tanto ambas variables toman valores enteros positivos.
- $y[p,t]$: Variable auxiliar que determina si el inventario de un producto en el día, ha bajado o no del valor $smin$ (usaremos como variable binaria que determina si hacemos o no un pedido, que nos llegará al día siguiente).
- $w[p,t]$: Variable auxiliar que determina si el stock de un producto en un día determinado alcanza el valor de máximo de carga asociado (por la modelización del problema, controlará si al acabar el día $stock[p,t] + DEM[p,t] = smax[p]$).
- $z[s,t]$: variable auxiliar binaria que determina si hemos realizado un pedido a un proveedor.

- **Parámetros:**
  - $DEM[p,t]$: ventas/demanda de cada producto en cada día.
  - $STOCKINI[p]$: inventario inicial de los productos.
  - $C_p[p]$: coste de compra de cada producto (uniforme en el tiempo).
  - $V[p,t]$: precio de venta de un producto en un día determinado.
  - $C_s[s]$: coste por pedido en cada proveedor.
  - CAP_{tienda}: capacidad de almacenaje de la tienda.
  - $PED_{max}[t]$: Nº de pedidos máximos a realizar por día.
  - $ICM[p]$: Imagen marca comercial a mantener por producto (nos determina un valor del que no puede bajar el stock).
  - $SUP[s,p]$: Matriz que contiene la información respectiva al suministro de los productos, tomando el valor 1 en la entrada (s,p) si el proveedor suministra ese producto y 0 en caso contrario.

A continuación definimos la expresión de la función objetivo y de las restricciones del problema:

Función objetivo:

$$min : \sum_{t} \bar{C}_p[p] \cdot cap_{hol}[t] + \sum_{t,s} z[s,t] \cdot C_s[s] + \sum_{p,t} (V[p,t] - C_p[p]) \cdot stock_{rup}[p,t] \quad (2.1)$$

Cada sumando está asociado a un coste de los que hemos presentado con anterioridad:

- Coste 1: $\sum_t \bar{C}_p[p] \cdot cap_{hol}[t]$ (ponderamos la variable cap_{hol}, que mide la cantidad de producto sobrante en cada día, por la media de los costes de los productos).
- Coste 2: $\sum_{t,s} z[s,t] \cdot C_s[s]$ (la variable binaria z nos da la información referente a los pedidos de los distribuidores, como el parámetro C_s contiene el coste asociado a cada proveedor, la expresión recoge el coste asociado a los pedidos).

- Coste 3: $\sum_{p,t}(V[p,t]-C_p[p])\cdot stock_{rup}[p,t]$ (los beneficios que nos reporta cada venta viene determinado por la diferencia entre el precio de venta y el de compra, como la variable $stock_{rup}[p,t]$ mide la demanda que no satisfacemos, debemos ponderar la perdida como el beneficio que dejamos de ganar).

Restricciones:

- $stock[p,0] = STOCKINI[p]$: imponemos que la variable que mide el inventario en el tiempo inicial tome el valor del parámetro de que recoge la cantidad de inventario inicial.
- $stock[p,t+1] \geq stock[p,t] - DEM[p,t+1]$: El inventario puede tomar dos valores: $smax[p] - DEM[p,t+1]$ o $stock[p,t] - DEM[p,t+1]$ (en ambos casos se le resta la demanda del día, porque las mediciones del stock se realizan al final del día), y por lo tanto, se da en general la restricción planteada.
- $stock[p,t+1] \leq stock[p,t] - DEM[p,t+1] + M \cdot y[p,t]$: esta restricción (junto con la anterior) impone $stock[p,t+1] = stock[p,t] - DEM[p,t+1]$ en el caso de que y=0. En el caso de realizar pedido, la M nos ayudará a que esta restricción no sea influyente (podemos pedir todo lo que queramos).
- $M \cdot y[p,t] \geq smin[p] - stock[p,t] + 1$: esta restricción impone que si $smin \geq stock$, entonces $y = 1$.
- $M \cdot (1 - y[p,t]) \geq stock[p,t] - smin[p]$: esta restricción impone que si $stock > smin$, entonces $y = 0$.
- $stock[p,t] \geq ICM[p] - stock_{rup}[p,t]$:determinamos el valor de la variable de holgura como la demanda/venta que dejamos de abastecer, ya que, en la práctica, el inventario no baja del ICM. En este caso, como la variable $stock_{rup}[p,t]$ es entera no negativa, la restricción solo tiene holgura hacia valores menores que ICM, no mayores.
- $stock[p,t] \leq smax[p] - DEM[p,t]$: el inventario no puede superar el máximo menos la demanda de ese día, ya que estamos suponiendo que nos llegan los pedidos al inicio del día y las mediciones de stock se realizan al finalizar el día (aunque al inicio de la mañana se recargue hasta smax, como medimos la final del día, necesitamos tener en cuenta la demanda).
- $\sum_p (stock[p,t] + DEM[p,t]) \leq CAP_{tienda} - cap_{hol}[t] \ \ t \in \{1,...,T\}$: esta restricción nos permite definir el valor de la $cap_{hol}[t]$ para valor de t, es decir, tenemos en cuenta si la capacidad de la tienda se sobrepasa o no, y en qué cantidad.
- $\sum_s z[s,t] \leq ped_{max}[t]$: limitamos el número máximo de pedidos que la tienda puede gestionar por día.
- $M * z[s,t] \geq SUP[s,p] \cdot y[p,t]$: si algún $y[p,t] > 0$, esta restricción obliga a $z = 1$.
- $z[s,t] \leq SUP[s,p] \cdot y[p,t]$: si todas las $y = 0$, esta restricción obliga a $z = 0$.
- $w[p,t] \geq stock[p,t] - (smax[p] - DEM[p,t])$: esta restricción impone que si $stock = smax - dem$, entonces $w = 1$.
- $M(1 - w[p,t]) \geq (smax[p] - DEM[p,t]) - stock[p,t]$: esta restricción impone que si $stock < smax - dem$, entonces $y = 0$.

- $w[p, t+1] \geq y[p, t]$: Siempre que tenemos un pedido sabemos que al día siguiente llenamos el inventario hasta el valor MAX. Esta restricción nos determina esta relación.
- $y[p, t] \in \{0, 1\}$: variable auxiliar de la realización del pedido.
- $w[p, t] \in \{0, 1\}$: variable auxiliar que mide si el inventario ha alcanzado el máximo.
- $z[s, t] \in \{0, 1\}$: variable auxiliar de pedido por proveedor.
- $stock[p, t] \in \mathbb{R}$: las variables de inventario son continuas.
- $smin[p], smax[p] \in \mathbb{N}$: las variables de control tienen que ser enteras.

2.1.2. Aproximación al problema de optimización

Una primera implementación de la modelización presentada no consiguió encontrar ninguna solución en un tiempo razonable de cómputo. Esto nos llevó a explorar modificaciones manuales que permitieran reducir el tiempo de cálculo. En esta sección intentamos presentar estos 'trucos' utilizados.

Cortes región factible

La primera idea que exploramos fue la de introducir cortes en la región factible. Uno de los cortes explorados fue:

- $min \in \{STOCKINI; ICM\}$.
- $max \in \{STOCKINI; \frac{CAPMAX}{2}\}$.

Este enfoque presentó mejoras en el rendimiento para tamaños del problema reducidos a 10 productos, pero siguió sin encontrar ninguna solución del problema con el tamaño objetivo en un tiempo de cómputo razonable.

Introducción de la solución inicial

Otra técnica ampliamente utilizada es la de iniciar el solver con una solución factible. Como contamos con una solución base con la que comparamos todas las soluciones, probamos a añadir esta solución como solución inicial del problema.

No obtuvimos mejora en el rendimiento del proyecto global (no encuentra solución en un tiempo superior a 1 h), entendemos que la cantidad de restricciones que genera la modelización para un tamaño de 1000 productos hace que el solver no sea capaz de comprobar la factibilidad de la solución inicial.

Permitirle GAP al solver

Otra opción que nos permiten los solver es fijar un GAP que aceptamos como criterio de parada. Exploramos esta opción, hasta el punto de aceptar la primera solución que el solver encuentra (GAP todo lo alto que quieras). Cuando el problema escala al tamaño objetivo, no se encuentra solución factible en el tiempo que fijamos como máximo. Lo cual era de esperar cuando ya hemos comprobado que en 1 h el solver no es ni capaz de comprobar si la factibilidad de una solución inicial.

Aun así, para problemas limitados en tamaño, la mejora en términos de velocidad para encontrar soluciones es importante. Por ello, las soluciones y reflexiones de los siguientes

apartados fueron exploradas con GAP de entre 30-50 %, ya que nos permitió mejorar claramente en términos de velocidad de cómputo.

División de la capacidad máxima

Observando que incluirle un GAP al solver para problemas reducidos de tamaños de 10-30 productos, mejoraba considerablemente la velocidad de ejecución. Esto nos sirvió de motivación para explorar soluciones que consistieran en dividir el problema global. En este caso, las restricciones que convierten al problema en global e introducen interacción entre los productos son, por un lado, la capacidad máxima de la tienda y, por otro, el pertenecer a proveedores distintos y compartir costes de envío.

Hemos explorado el optimizar por separado los productos de cada proveedor, asignándole de forma igualitaria la capacidad máxima a cada proveedor. Por cómo hemos definido nuestro problema, contamos con un total de 20 proveedores que gestionan de forma exclusiva un catálogo de 50 productos cada uno. Por ello, al realizar esta división, contamos con 20 problemas de optimización (uno por proveedor) con un tamaño de 50 productos por problema.

El experimento devolvió soluciones aceptables en términos de optimalidad (respecto al objetivo base), pero el tiempo de cómputo pasaba por mucho el objetivo máximo de 1 h establecido.

Después del fracaso de la división por proveedores, el siguiente foco fue intentar reducir aún más el tamaño de los subproblemas (en términos de cantidad de productos). El bajar a nivel producto, eligiendo al azar conjuntos de 10 productos (que conocíamos que podía hacer el problema asequible en tiempo) tenía el problema de que perdemos gran parte de la capacidad de optimización de los pedidos.

Aun así, exploramos esa vía, dividiendo los 1000 productos en 100 grupos de 10 productos y asignando a cada grupo un 1 % de la capacidad máxima y realizando los 100 problemas de optimización pertinentes. Los resultados en términos de optimalidad fueron malos y por eso no se siguió explorando esta idea.

2.1.3. Genético

El mal rendimiento computacional de las diferentes formulaciones estudiadas hasta este momento hizo que nos decidiéramos a probar heurísticas. En este caso, presentaremos la definición teórica de los algoritmos genéticos y cómo hemos adaptado su funcionamiento al problema que venimos trabajando.

Teoría

Los algoritmos genéticos basan sus ideas en los procesos de la evolución biológica. Entre las ideas claves a rescatar de estos procesos biológicos se encuentran la recombinación genética y la selección natural. Expliquemos los mecanismos biológicos que justificarán los pasos del algoritmo.

- **Selección**: En los ecosistemas naturales, los individuos con características genéticas favorables presentan mayores opciones de sobrevivir y reproducirse.
- **Reproducción**: El proceso de reproducción a nivel cromosómico consiste en la combinación de material genético de los progenitores.
- **Mutación**: Las mutaciones genéticas son procesos que ocurren con probabilidades reducidas que consisten en cambios aleatorios de parte del material genético.

En el caso de la optimización, estas ideas tendrán su utilidad si caracterizamos las soluciones factibles de nuestro espacio como individuos y en consecuencia le asignamos una definición genética (por ejemplo, la expresión vectorial de la solución en el espacio de búsqueda). El algoritmo partirá de una población inicial que se irá modificando mediante la aplicación de los procesos biológicos antes mencionados. Cada iteración del algoritmo se definirá como una generación, en analogía a los términos biológicos. En la figura 2.1 podemos ver un resumen de las fases que tiene la determinación de una generación.

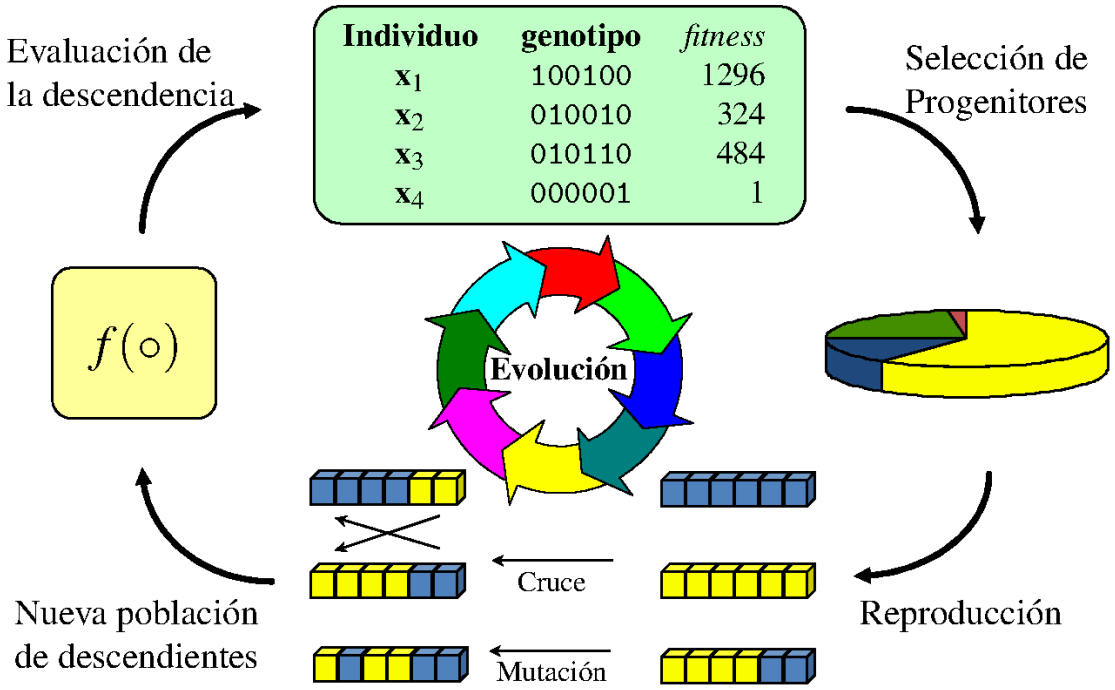

Figura 2.1: Resumen del proceso iterativo de un algoritmo genético. Imagen tomada de: https://images.app.goo.gl/sNYbgNbre2XYEhqU7 Enlace

Definimos a continuación los pasos de los que consta cada generación:

- **Población inicial**: Cada iteración parte de una población inicial, un conjunto de puntos del espacio de búsqueda del problema de optimización. Además, todos los puntos han sido evaluados previamente en la función objetivo y, por tanto, podemos conocer cuáles son los mejores individuos.

- **Selección**: Este proceso consiste en la selección de los mejores individuos de nuestra población. Para ello se define un proceso 'competitivo' entre individuos de la población. Lo habitual es utilizar una modalidad de torneo. Se selecciona un número (parámetro del modelo) de muestras, que puede ser, con remplazo o no, de la población inicial. Después, dentro de cada muestra, el 'ganador' será el individuo que presente un mejor rendimiento basado en la función de optimización que hemos fijado.

- **Reproducción**: En este caso, se fija de antemano la probabilidad con la cual 2 individuos se reproducen. Entonces se emparejan los individuos y, de forma aleatoria, algunos de ellos intercambian parte de la materia genética, que en este caso se traduce en el intercambio de valores de algunas coordenadas.

- **Mutación**: Otro parámetro definido por el usuario es la probabilidad de mutación. Se aplica una modificación aleatoria en alguna de las coordenadas del individuo respetando la probabilidad prefijada.

- **Evaluación**: El último paso es volver a evaluar a todos los individuos de la nueva población para estar en las mismas condiciones en que habíamos empezado la iteración.

Como todo proceso iterativo, necesitamos fijar un criterio de parada. En este caso, en cada generación se extrae la evaluación del mejor individuo y se guarda ese valor como la mejor solución que se ha encontrado hasta ese momento. El usuario puede definir un criterio de parada basado en el número de generaciones que el algoritmo no consigue encontrar un mejor individuo.

Adaptación del problema

En este apartado explicaremos cómo hemos definido los diferentes elementos del algoritmo genético para obtener una solución de nuestro problema.

- **Individuo**: Hemos caracterizado una solución (o individuo) como la concatenación de los valores de MIN-MAX. Es decir, cada elemento de la población será un vector de dimensión 2000, donde los primeros 1000 elementos recogen la información del valor MIN de los 1000 productos y las siguientes 1000 posiciones los valores de MAX de los mismos 1000 productos.

- **Población inicial**: Como línea conductora de los retos más clave que nos hemos enfrentado en el proyecto, vuelve a aparecer el problema de cómputo y cómo debemos restringir nuestras soluciones a procesos que no sobrepasen 1 h de cálculo.

  En este caso, esta restricción nos llevó en un inicio a limitar la población inicial a un total de 100 individuos. Como es evidente, una muestra de 100 puntos en un espacio con un tamaño de 2000 dimensiones es claramente insuficiente para capturar la variabilidad. Es por ello que no hemos podido utilizar una población inicial elegida de forma aleatoria, porque comprobábamos que en una hora de cómputo la solución aportada por el genético no mejoraba ni la solución del modelo base en términos de optimalidad.

  Para solventar este problema, optamos por definir una población inicial ad-hoc, donde buscamos aplicar nuestro conocimiento de negocio para generar individuos (valores MIN-MAX) que conocemos a priori que presentarán mejor optimalidad (en general) que las soluciones aleatorias.

  Esta idea nos llevó a desarrollar un algoritmo que define los individuos de la población inicial. Expliquemos el proceso para un individuo en concreto, aunque este proceso se replica 100 veces para obtener toda nuestra población inicial. La idea general detrás de esta solución es realizar algo similar a la construcción del modelo base, pero introduciendo cierta variabilidad tanto en los valores MIN como en MAX.

  - **Generación de valores MIN**: Cada individuo de la población es un vector que contiene 1000 valores MIN, uno para cada producto de nuestro estudio. Pues para cada uno de esos productos, se realiza el cálculo del día de la ruptura de inventario (como en el caso del modelo base). Una vez calculado este valor, podemos conocer el inventario tanto el día previo a la ruptura, como el de 2

días previos (puntos rojos en la figura 2.2). Para cada producto, el valor MIN se define mediante la selección aleatoria de uno de los dos valores anteriores.

- **Generación de valores MAX**: Conocemos el valor MIN seleccionado para cada producto. El siguiente paso consta en la generación de un valor aleatorio entre [1,7] para cada producto. Este parámetro aleatorio representa el número de días de demanda a futuro que vamos a acumular. El valor de MAX vendrá determinado por el valor de MIN más la demanda acumulada que se ha calculado dependiente de ese parámetro aleatorio (todos los puntos verdes de la figura 2.2 representan las posibilidades que tiene el valor MAX para un único producto).

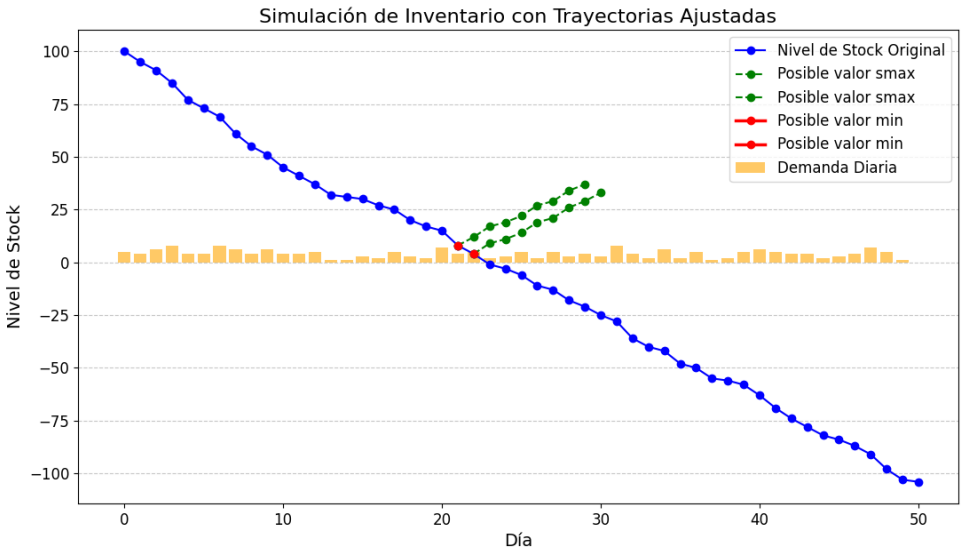

Figura 2.2: Proceso generación población inicial

En la gráfica de la figura 2.2 podemos observar que para cada producto el algoritmo selecciona un valor MIN-MAX de un total de 14 combinaciones. Consideramos esta variabilidad suficiente para abordar el proceso de optimización con el algoritmo genético, siendo conscientes de que la variabilidad introducida en la población inicial limitará el campo de búsqueda de nuestro algoritmo.

- **Selección y evaluación**: El proceso de selección necesita la definición de la función objetivo que evalúa a cada individuo. En este caso, no trabajamos con una función matemática explícita. Para cada solución hacemos una simulación de la evolución real del stock en el período de 14 días, donde computamos todos los costes mencionados anteriormente (incluido el tirado de producto). Este simulador es el que se reutilizará para comparar finalmente las soluciones propuestas en el periodo de evaluación. Una vez evaluados los individuos, se propuso un proceso de selección basada en un torneo de 3 individuos (seleccionados de forma aleatoria y con repetición) donde el mejor evaluado es seleccionado para la próxima generación. Este proceso de torneo se repite 100 veces para no disminuir la cantidad de individuos de la población entre generaciones.

- **Reproducción y Mutación**: Se fijaron como probabilidad de reproducción el 50 %

y un 20 % de mutación.

- **Criterio de parada**: Hemos fijado 2 criterios de parada simultáneos. El primero se basa en un criterio sobre la solución óptima, cuando transcurren 20 generaciones sin encontrar mejoras en la solución, se para el proceso del genético. La otra condición está relacionada con el objetivo de tiempo máximo que le permitimos al algoritmo para encontrar una solución, que en este caso se fijó de antemano como un máximo de 1 h.

  El algoritmo genético usa el simulador del inventario de la tienda. Este algoritmo sufrió mejoras en el desarrollo del proyecto. En el momento en que se fijó el tamaño de la población inicial en 100 individuos, el criterio de parada de tiempo máximo se accionaba a menudo debido a una implementación poco optimizada de este simulador. En las últimas versiones del simulador de inventario, se mejoró mucho la velocidad de cómputo debido a las mejoras de rendimiento introducidas (optimización de código). Finalmente, una ejecución del genético toma de media unos 15 minutos en satisfacer el criterio de parada basado en la optimalidad de la solución encontrada.

Notas técnicas sobre la implementación

En la redacción del trabajo se ha decidido excluir el contenido que contenga código. Aun así, consideramos interesante mencionar en el caso del genético las características y parámetros que contiene la librería utilizada para la ejecución del mismo.

Para la implementación del algoritmo genético se ha empleado la librería *DEAP (Distributed Evolutionary Algorithms in Python)* [3]. Esta librería ha sido diseñada con un enfoque modular que permite definir y personalizar con facilidad distintos tipos de algoritmos evolutivos, entre ellos algoritmos genéticos.

DEAP es una librería que basa su estructura en clases y objetos predefinidos que representan los elementos fundamentales de los algoritmos evolutivos: individuos, poblaciones, operadores de selección, cruce y mutación, función de evaluación. Esta organización permite adaptar los algoritmos a diferentes problemas de optimización sin necesidad de modificar el núcleo de la librería.

Este enfoque permitió poder abordar el uso de genéticos sin tener que implementar desde cero los mecanismos evolutivos fundamentales. Nos hemos limitado a la personalización del problema de optimización que estamos abordando.

Los objetos definidos en la clase hacen referencia a todos los elementos antes presentados de forma teórica. Presentemos brevemente los elementos que se han utilizado (de forma personalizada):

- **creator**: Permite definir los individuos de forma estandariza, en conjunto con la función de evaluación.

- **toolbox**: Es el módulo central donde están implementados todos los operadores evolutivos que se pueden utilizar, en nuestro caso: cruce, mutación y selección.

- **parámetros clave**: La librería contiene parámetros configurables del algoritmo como: tamaño de la población, número de generaciones, probabilidad de cruce, probabilidad de mutación...

Otro punto clave a destacar de esta librería es que incluye un soporte para paralelización, lo que permitió acelerar la ejecución de los problemas de optimización abordados.

2.2. Predicción

Para la etapa de predicción estábamos buscando modelos que permitieran gestionar diferentes características que encajen con los objetivos de negocio planteados. La primera era poder gestionar muchas series temporales de forma global debido a que posibles requerimientos por parte de clientes reales pueden presentar problemas de volumetría de datos. Otra de las características que consideramos imprescindibles es la posibilidad de realizar predicciones multi temporales, ya que necesitamos estimar la demanda a 14 días vista para poder realizar nuestro proceso de optimización. También, viendo el dataset con el que contamos y suponiendo que en futuros proyectos es probable contar con variables de contexto externas a las propias series, otro de los requisitos que planteamos fue el poder añadir esta información en nuestros modelos.

Finalmente, se decidió optar por 2 tipos de modelos de Aprendizaje Profundo, como son DeepAR y TFT (Temporal Fusion Transformers), que tienen estructuras diferentes, pero presentan características similares a las necesidades planteadas. A continuación, presentaremos brevemente qué es el Aprendizaje Profundo, y añadiremos una breve descripción de los modelos seleccionados. La explicación será concisa, ya que se considera más relevante la aplicación práctica del proyecto y la arquitectura diseñada para llevar a cabo los procesos de predicción.

2.2.1. Aprendizaje Profundo

El Aprendizaje Profundo se clasifica como una rama de la inteligencia artificial, debido a que otorga a la máquina las capacidades necesarias para realizar tareas que habitualmente requieren de inteligencia humana.

El Aprendizaje Profundo ha tenido una gran implantación en tareas como reconocimiento de imágenes, procesamiento de lenguaje natural, conducción autónoma, etc. Sus características le permiten modelar patrones complejos a partir de grandes volúmenes de datos.

Para abordar el concepto de Aprendizaje Profundo, necesitamos presentar algunas definiciones previamente:

Definición 2.1 (Neurona)**.** Una **neurona** es la unidad básica de procesamiento en una red neuronal artificial. Recibe un conjunto de entradas numéricas, cada una ponderada por un peso asociado, y computa una combinación lineal de estas entradas. A continuación, aplica una función de activación no lineal para generar su salida. Matemáticamente, una neurona realiza la operación:

$$y = \phi\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

donde x_i son las entradas, w_i los pesos, b el sesgo (bias), y ϕ la función de activación. Las neuronas se organizan en capas y forman conjuntamente redes que permiten modelar funciones complejas.

Definición 2.2 (Capa)**.** Una **capa** en una red neuronal es una unidad funcional compuesta por un conjunto de neuronas o nodos que reciben una entrada, aplican una transformación (usualmente lineal seguida de una función de activación no lineal) y generan una salida. Las capas pueden clasificarse según su función: capa de entrada, capas ocultas y capa de salida. Cada capa transforma su entrada en una representación intermedia que facilita la extracción progresiva de características en tareas de aprendizaje automático.

Definición 2.3 (Red neuronal)**.** Una **red neuronal** es una arquitectura computacional compuesta por múltiples capas conectadas entre sí, donde cada capa transmite su salida como entrada a la siguiente. Su objetivo es aproximar funciones complejas mediante un proceso de aprendizaje supervisado o no supervisado. Las redes neuronales profundas (deep neural networks) se caracterizan por tener múltiples capas ocultas, lo que les permite capturar relaciones jerárquicas y patrones complejos en los datos.

El término profundo proviene de su estructura de capas, las cuales están constituidas de parámetros que la red aprende mediante procesos de optimización. Estas capas transmiten la información de forma secuencial, transformando los datos del input en un output de una forma no explicable (o no interpretable). Podemos ver en la figura 2.3, cómo se construye esta estructura de capas:

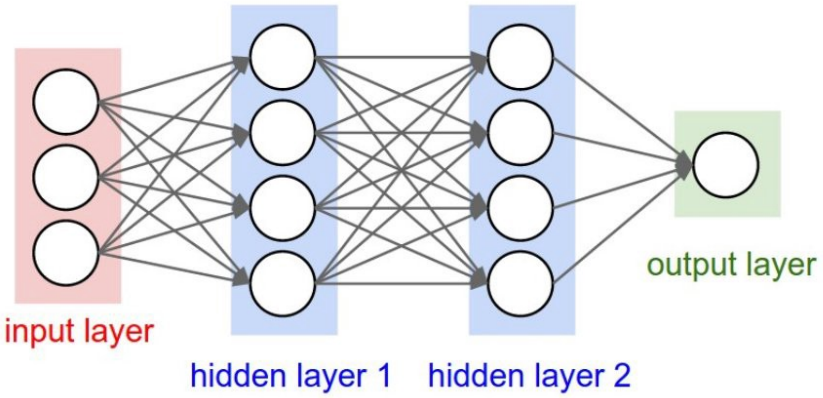

Figura 2.3: Esquema red neuronal

TFT

El primero de los modelos de Aprendizaje Profundo seleccionado es TFT [4]. Este modelo cumple los requisitos expuestos en la introducción, ya que permite la gestión de predicciones multi-horizonte y la gestión de variables de contexto. Además, es un modelo interesante, ya que está basado en la predicción cuantílica. En nuestro caso utilizaremos la predicción mediana, pero tenemos en cuenta que esto permitiría gestionar una mayor cantidad de información en futuros refinamientos del proceso de optimización.

Como se menciona en la introducción, el objetivo de esta explicación no es centrarse en las complicadas arquitecturas de las soluciones usadas. Sin embargo, es importante señalar que TFT está basado en un mecanismo de atención, una técnica ampliamente conocida y utilizada en modelos avanzados de procesamiento de lenguaje natural, como los conocidos GPT (Generative Pre-Trained Transformer). La idea principal de esta metodología es crear modelos que se enfoquen dinámicamente en las partes más relevantes de la entrada, reduciendo la dimensionalidad de los embeddings iniciales mediante proyecciones en subespacios latentes, y capturando relaciones temporales y contextuales relevantes.

DeepAR

El segundo de los modelos seleccionados es DeepAR [5]. Presenta muchas similitudes con el TFT debido a las características que buscábamos para nuestra solución. En este caso, el output del modelo es una predicción probabilística (aunque finalmente realizáramos el

proceso de optimización con la demanda determinística dada por la media). En este caso, se introducen todas las series temporales ($z_{i,t}$) con el objetivo de tener un modelo global que predice la siguiente distribución condicionada:

$$P\left(z_{i,t_0:T} \mid z_{i,1:t_0-1},\, x_{i,1:T}\right) \tag{2.2}$$

En este caso, estaríamos prediciendo la probabilidad condicionada de cada serie futura $[z_{i,t_0}, z_{i,t_0+1}, ..., z_{i,T}]$ dado los valores de las series temporales en el pasado (periodo de entrenamiento) $[z_{i,1}, ...z_{i,t_0-1}]$ donde el t_0 denota el tiempo a partir del cual vamos a realizar la predicción. Además, como vemos en la expresión anterior, nos permite introducir variables de contexto $x_{i,1:T}$ que pueden ser característicos del individuo (ejemplo: es un subproducto lácteo) o una característica temporal (ejemplo: mes del año, festividades, ...).

De nuevo, la estructura de la red neuronal no es parte del marco del trabajo, pero sí queremos indicar que el modelo se basa en la factorización de una función de verosimilitud basada en una estructura de autocorrelación recurrente. Bajo el supuesto de que los datos siguen una distribución paramétrica (parámetro del modelo), el algoritmo se encarga de estimar los parámetros de esa distribución paramétrica que va a usar para devolver las predicciones probabilísticas de nuestra serie temporal.

Notas técnicas sobre la implementación

En la redacción del trabajo se ha decidido excluir el contenido que contenga código. Aun así, consideramos interesante mencionar en el caso de los modelos de series temporales las características y capacidades que ofrece la librería utilizada en nuestro proyecto.

Para la implementación de los modelos DeepAR y TFT se ha utilizado la librería *GluonTS (Gluon Time Series)* [6], desarrollada por Amazon. GluonTS es una librería de código abierto enfocada en el modelado y la predicción de series temporales, que proporciona una colección de modelos estadísticos y de Aprendizaje Profundo integrados con los frameworks Apache MXNet y PyTorch.

Uno de los principales valores de GluonTS es su diseño modular, que permite separar claramente los siguientes componentes: definición del modelo, preprocesamiento de datos, entrenamiento y evaluación. Esto facilitó el uso de distintos modelos con un pipeline unificado y reproducible, que era una característica imprescindible para poder llevar a cabo el objetivo del proyecto en el tiempo propuesto. El uso de esta librería hace que el proceso de entrenamiento de modelos se base en la personalización de ciertos objetos estandarizados en la librería (distribuciones predefinidas). También contamos con una amplia variedad de parámetros que nos han permitido configurar los procesos de entrenamiento en cada modelo. Como la optimización de parámetros será uno de los pasos claves de nuestro pipeline global, vamos a mencionar aquí qué parámetros hemos personalizado en cada modelo.

En el caso de TFT manejaremos los siguientes parámetros:

- **hidden_dim**: Hace referencia al número de unidades de capas ocultas (ver Figura 2.3). Un mayor valor permite capturar relaciones más complejas, pero en contrapartida aumenta el coste computacional.
- **context_length**: Este parámetro recoge el número de pasos del pasado que el modelo va a utilizar como entrada para predecir el futuro.
- **dropout_rate**: Es la tasa de desactivación de neuronas durante el entrenamiento y tiene el objetivo de reducir el sobre ajuste. Este parámetro toma valores entre 0 y 1, siendo un valor de 0.1 una reducción de un 10 % en la activación de las neuronas en cada iteración.

- **lr**: Es la tasa de aprendizaje del optimizador.

Para entrenar DeepAR usaremos los siguientes parámetros:

- **num_layers**: Hace referencia al número de capas de recurrentes.
- **hidden_size**: Hace referencia a las capas ocultas de la arquitectura.
- **context_length**: Cantidad de pasos que toma la ventana del historial que sirve como entrada para predecir el futuro. Esto determina la cantidad de información pasada que usa el modelo en cada predicción.
- **dropout_rate**: Es la tasa de desactivación de neuronas durante el entrenamiento y tiene el objetivo de reducir el sobre ajuste. Este parámetro toma valores entre 0 y 1, siendo un valor de 0.1 una reducción de un 10 % en la activación de las neuronas en cada iteración.
- **lr**: Es la tasa de aprendizaje del optimizador.

Además de los parámetros antes mencionados, existen otros elementos clave de la librería que se han usado en la configuración y entrenamiento de los modelos definidos:

- **Estimator**: es el objeto que permite configurar y construir el modelo, estableciendo hiperparámetros como el número de épocas, tamaño de ventana de entrada, tipo de distribución de salida, etc.
- **Predictor**: una vez entrenado, este objeto encapsula el modelo listo para realizar predicciones sobre nuevas series temporales.
- **Train/Evaluation Datasets**: GluonTS define estructuras de datos específicas que permiten dividir las series temporales en conjuntos de entrenamiento y validación con ventanas móviles o fijas.
- **Distribución de salida**: GluonTS permite seleccionar entre diferentes distribuciones para modelar la variable objetivo. En nuestro caso, se emplearon distintas distribuciones en función del modelo (por ejemplo, Tweedie e Implicit Quantile Network para DeepAR; Quantile y Binomial Negativa para TFT).

Otra funcionalidad relevante de GluonTS es su compatibilidad con entornos paralelos y su integración con GPUs, lo que permite escalar eficientemente los procesos de entrenamiento cuando se trabaja con grandes volúmenes de datos o arquitecturas profundas.

2.2.2. Arquitectura

En esta sección explicaremos la arquitectura usada en la solución técnica del proceso de predicción. Tenemos un proceso en 3 etapas: Preprocesamiento, Entrenamiento y Evaluación, donde 2 de ellas difieren entre enfoques, por ello intentamos destacar en ambos casos cajas separadas en esos pasos del proceso global. Sin embargo, el proceso de entrenamiento es común a ambos. El esquema de la Figura 2.4 intenta trasladar esta visión.

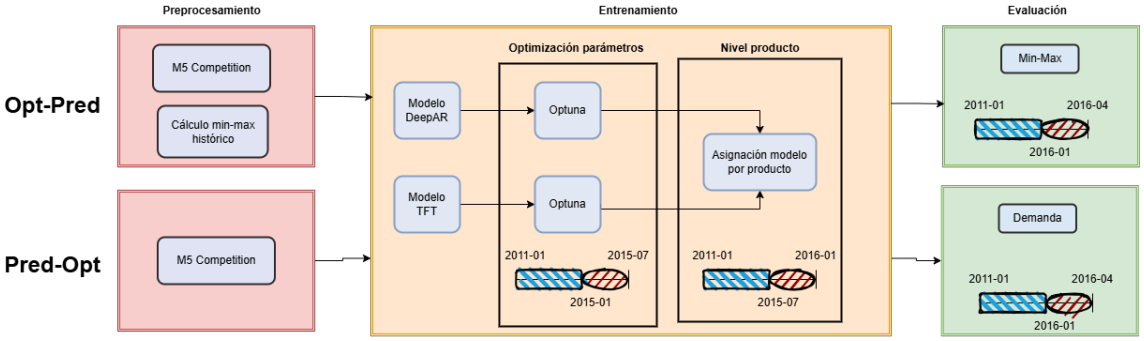

Figura 2.4: Flujo de trabajo predicción

Preprocesamiento

Entendemos que esta arquitectura fue creada para la parte de predicción. Por eso debemos diferenciar previamente que la granularidad es distinta en ambos procesos:

- **Optimización-Predicción**: Partimos de un proceso previo de optimización en ventanas de 14 días sobre el histórico de datos. Es por eso que nuestros datos de entrenamiento tienen una granularidad de 14 días (importante al introducir variables de contexto) que será la misma granularidad del output (que en este caso predice solo un paso hacia delante, es decir, el próximo MIN-MAX).
- **Predicción-Optimización**: En este caso partimos del conjunto de datos completo, que recoge datos de ventas de forma diaria y otras variables de contexto a nivel diario también. El output será la demanda a nivel diario, pero necesitaremos realizar la predicción de la ventana de 14 días a futuro.

El proceso de preprocesamiento tiene un peso mucho mayor en el Optimización-Predicción, ya que necesitamos realizar un cálculo de histórico de MIN-MAX sobre los datos que tenemos, este proceso es el que ha limitado el tiempo de cómputo del algoritmo genético, ya que tuvimos que realizar ese proceso en ventanas de 14 días sobre un histórico de más de 5 años. Teniendo estos valores construidos, la parte fundamental del proceso de preprocesamiento está completada.

La otra parte de este proceso, fue el adecuar qué variables de contexto podemos añadir en cada caso en los modelos de predicción. Usaremos una imagen (ver figura 2.5 que hace referencia al tipo de variables que puede utilizar el TFT, pero que tiene una estructura similar en DeepAR y nos permite esquematizar qué datos podemos introducir en el modelo.

Trabajaremos con 4 tipos de datos en el modelo predictivo. A continuación, realizamos una breve caracterización de cada uno:

- **Past targets:** Hace referencia a la variable de estudio (o que se intenta predecir).
- **Observed Inputs:** Son variables de contexto que solo conocemos en el pasado. En nuestro caso, contamos con el dataset *calendar* que recoge festividades y eventos que pueden tener influencia en las ventas (final NBA, Navidades, Ramadán, ...).
- **Known Inputs:** Son variables que se conocen de antemano, tanto en el pasado como en el futuro. El ejemplo común es el calendario (día de la semana, mes, año).
- **Static Covariates**: Son variables que no cambian con el tiempo. En nuestro caso de estudio, la tipología del producto puede servir de ejemplo.

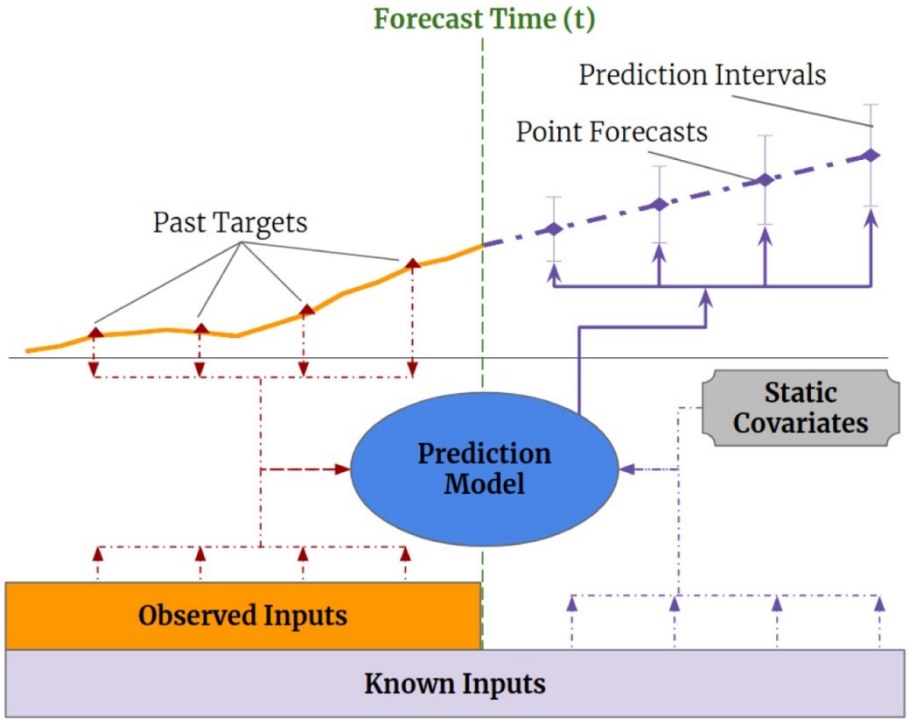

Figura 2.5: Esquema datos entrenamiento

Una vez hemos introducido la potencia con la que cuentan ambos algoritmos a la hora de aprovechar información extrínseca a las series temporales, tenemos que presentar cómo hemos trabajado esa información para en cada enfoque.

Empezamos por el Predicción-Optimización. En el enfoque clásico, la *Past target* son las ventas (que utilizamos para predecir la demanda futura). Para los *Observed Inputs* tenemos la siguiente lista resumen:

- **event_type**: Es una variable indicadora, que nos dice si en un día hay un registro de algún evento importante.
- **event_name**: Es la variable que pone nombre al evento que se indica en el type.
- **Eventos prolongados**: Algunas de las entradas anteriores indicaban el inicio y el final de una festividad, tomando la descripción event+Inicio y event+Final. Es por eso que formulamos nuevos campos sintéticos, donde tomará valor 1 todo el periodo de festividad (esto es necesario para poder diferenciar los eventos únicos, de los eventos que tienen duraciones mayores). Los campos sintéticos creados fueron los siguientes: NBA, Ramadan, Chankah, Pesach, Lent, Halloween.
- **Snap**: Esta variable es una indicadora sobre los programas de ayudas para gente con bajos recursos. El valor 1 representa que el programa estaba activo en ese momento.

Otro tipo de información que podemos introducir es la que definíamos como *Static Covariates*, en este caso, para cada serie (un producto) conocíamos el departamento y categoría que estaba asociado. Por último, están los *Kwon Inputs* que en este caso hacen referencia a campos o información temporal: fin de semana (0,1), mes, año, semana.

El enfoque no clásico, Optimización-Predicción, tiene como *Past target* el valor MIN o MAX (se trabaja en procesos separados). Esto implica que la granularidad de la serie

es bisemanal, esto provoca un problema a la hora de introducir la información de contexto que tenemos a una granularidad diaria. En el caso de los *Observed Inputs*, se han realizado agregados a nivel bisemanal sumando las variables indicadoras que teníamos a nivel diario. Este cambio realiza una agrupación de la información. Un caso práctico puede ser un período de 14 días donde la variable de NBA se transforma de un vector: $(0, 0, 0, 0, 0, 0, 0; 1, 1, 1, 1, 1, 1, 1)$ a 7. Esta fue la solución que se ha encontrado para intentar introducir la misma información en ambos casos (es obvio que el trabajarla de forma agregada implica una reducción de la información utilizada).

Entrenamiento

Una vez contamos con los datos, empezamos el paso de entrenamiento. Diferenciamos dos fases, en la primera trabajamos por separado entre varios modelos globales, y en la segunda fase bajaremos a nivel de producto, asignando a cada serie qué modelo se va a encargar de predecirla.

La primera fase la conocemos como optimización de parámetros, donde optimizamos los valores de los parámetros que hemos presentado anteriormente para cada modelo. Para ello usamos Optuna, que es un algoritmo de optimización bayesiano, al cual se le definen los parámetros a optimizar y una rejilla de valores de búsqueda para dichos parámetros. Optuna también necesita un criterio de minimización. El esquema presentado en la figura 2.6 resume esta información.

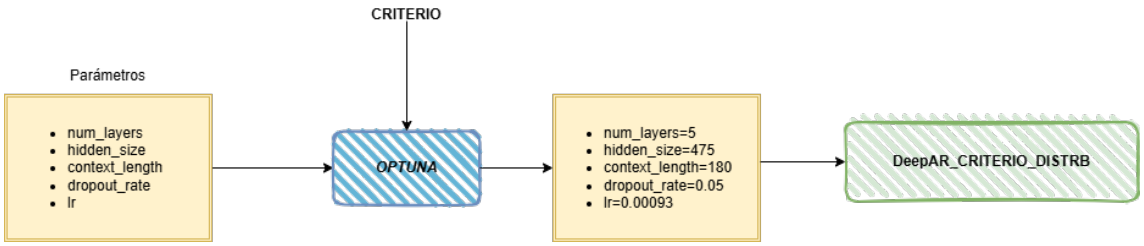

Figura 2.6: Optuna

El criterio, es un criterio de error sobre las predicciones (por eso necesitamos un conjunto de evaluación para este proceso de entrenamiento de parámetros distinto al conjunto de evaluación donde se compararán los costes).

Definición 2.4. Definimos el residuo en la predicción como $e_i = z_i - \hat{z}_i$.

Entre los posibles criterios de error hemos seleccionado los 3 siguientes:

- **Mean Absolute Error (Error Absoluto Medio)** que se calcula como sigue:

$$MAE = \frac{1}{N}\sum_{i=1}^{n} |e_i| \tag{2.3}$$

  Es un criterio de fácil interpretación, que se basa en el promediar los residuos en valor absoluto.

- **Mean Squared Error (Error cuadrático medio)** que se calcula como sigue:

$$MSE = \frac{1}{n}\sum_{i=1}^{n} e_i^2 \tag{2.4}$$

Este criterio se basa en el promedio de los errores al cuadrado, esto se traduce en una mayor penalización de aquellos errores que sean mayores.

- **Normalized Root Mean Squared Error (Raíz del Error Cuadrático Medio Normalizado)** que se calcula como sigue:

$$NRMSE = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2}}{\hat{z}} \tag{2.5}$$

En la expresión anterior, el término normalizador es la media de las predicciones ($\hat{z}$).

En la Figura 2.6 podemos ver que el Output toma el nombre DeepAR_CRITERIO_DISTRB. Esta construcción está determinada por los elementos que vamos a combinar a la hora de generar 12 modelos globales. El primer parámetro es el tipo de modelo (en el caso de la Figura 2.6 tenemos el ejemplo de parámetros de un modelo de DeepAR). El segundo de los elementos a combinar es el tipo de criterio de penalización que se usa en Optuna para la optimización de parámetros del modelo. El último elemento que introducimos es el tipo de penalización que usará el modelo de Aprendizaje Profundo a la hora de calcular los parámetros de la red (los nombres recogidos son clases de la propia librería de gluonts).

Nombre del modelo	**Algoritmo**	**Criterio**	**Distribución**
DeepAR_MAE_IQN	DeepAR	MAE	Implicit Quantile Network
DeepAR_MAE_Tweedie	DeepAR	MAE	Tweedie
DeepAR_MAPE_IQN	DeepAR	MAPE	Implicit Quantile Network
DeepAR_MAPE_Tweedie	DeepAR	MAPE	Tweedie
DeepAR_NRMSE_IQN	DeepAR	NRMSE	Implicit Quantile Network
DeepAR_NRMSE_Tweedie	DeepAR	NRMSE	Tweedie
TFT_MAE_Quantile	TFT	MAE	Quantile
TFT_MAE_NegBin	TFT	MAE	Binomial Negativa
TFT_MAPE_Quantile	TFT	MAPE	Quantile
TFT_MAPE_NegBin	TFT	MAPE	Binomial Negativa
TFT_NRMSE_Quantile	TFT	NRMSE	Quantile
TFT_NRMSE_NegBin	TFT	NRMSE	Binomial Negativa

Cuadro 2.1: Valores de los diferentes tipos de reparto

La tabla 2.1 recoge la definición de los 12 modelos globales que resultan de la combinación de los tres parámetros: algoritmo, criterio, distribución. Cada uno de estos modelos se

optimizará y entrenará de forma totalmente independiente y su capacidad de predicción será comparada en la siguiente etapa.

Podemos observar en la figura 2.4 que tenemos un segundo recuadro denominado 'Nivel de producto' en la cajita de entrenamiento. Este segundo paso, dentro de la fase de entrenamiento, se basa en una simple decisión, intentar asignar a cada serie temporal (producto) el modelo global que mejor predicción presente de esa serie. Para ello, reentrenamos el modelo en el dataset histórico anterior (entrenamiento+validación) y añadimos 6 meses más de nuestro histórico para realizar el test del error de cada modelo global a cada serie.

Una vez realizamos todas las predicciones a nivel producto con cada modelo global, podemos calcular las siguientes métricas: MAE, MAPE, NRMSE. Una vez calculadas las métricas de error de predicción, fijamos un criterio de asignación del modelo a nivel de producto basado en una votación de los 3 criterios anteriores. Esto se traduce en que si un modelo global presenta menos error en 2 métricas (o las 3) que el resto de modelos globales, ese modelo será el encargado de predecir ese producto. En caso de empate, es decir, que el modelo que minimiza cada criterio sea distinto, nos decantaremos por el modelo global que minimice el criterio MAE.

Queda finalizado este proceso cuando cada ID producto (serie temporal) tiene un modelo global que se encarga de realizar su predicción en la franja temporal en la que se busca comparar ambos enfoques, es decir, la franja de evaluación de todo el proyecto.

Evaluación

Volviendo sobre la Figura 2.6, podemos observar las cajitas temporales inferiores (iguales en cada caso). Eso indica como debemos tomar todo el histórico de datos que habíamos utilizado hasta el paso anterior, 01/2016 para realizar un último reentrenamiento de los modelos (por optimización en el uso de código todos los modelos globales se reentrenaron con todas las series temporales, no se particularizó el reentrenamiento solo con aquellas series que le han sido asignadas).

Una vez hemos reentrenado y conocemos qué modelo global se utilizará para predecir cada serie, realizamos un proceso de predicción individualizado (cada serie usa un modelo) en el período que se había fijado como test del trabajo (01/2016-05/2016).

Importante recordar los outputs diferenciados a nivel de enfoque:

- Optimización-Predicción: Devuelve directamente los valores MIN-MAX que se utilizarán en el simulador del cálculo del coste obtenido por esta solución (siendo esta predicción solo del periodo siguiente). Cabe destacar que este enfoque requiere de dos procesos de predicción distintos a nivel de MIN y de MAX, con lo cual tenemos que evaluar el doble de series temporales que en el otro enfoque.

- Predicción-Optimización: Devolvemos la demanda de los próximos 14 días a nivel de producto.

Capítulo 3

Comparación de enfoques

Desde un principio, hemos definido como objetivo principal del proyecto el comparar los dos enfoques. Un enfoque clásico de Predicción-Optimización, con un enfoque más inusual que intercambia el orden de las técnicas, para realizar una Optimización-Predicción. A continuación, añadiremos alguna imagen de los procesos de predicción, con el objetivo de justificar los resultados económicos obtenidos, que se expondrán en el último apartado:

3.1. Enfoque Predicción Optimización

Si hacemos memoria, este es el enfoque que denominábamos como clásico, donde realizábamos un proceso de predicción de la demanda de nuestros productos. Dicha información es la base que sustenta el proceso de optimización asociado a una gestión eficiente del inventario. Por lo tanto, nuestro proceso de predicción de series temporales utiliza el histórico de ventas de productos de la compañía.

Fijándonos en los pasos intermedios, hemos podido apreciar que existe una clara diferencia en el comportamiento de los modelos predictivos entre enfoques. Las series temporales de ventas (las que se utilizan en el Predicción-Optimización) han aportado mejores resultados en el proceso de predicción. Podemos suponer que esta mejora del rendimiento es debida a que estas series tienen, por un lado, mayor información (debido a una mayor granularidad) y a su vez mayor regularidad que las series que veremos en el otro enfoque.

Aun así, las series a nivel producto no dejan de estar muy desagregadas y presentan gran variabilidad que es difícil de capturar para cualquier modelo de predicción. La siguiente imagen contiene las diferentes predicciones realizadas por nuestros modelos (en etapas de 14 en 14 días) en el período que se utilizó como evaluación de los costes.

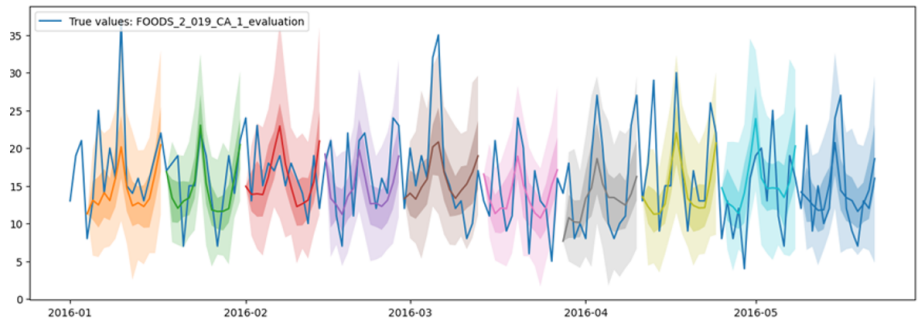

Figura 3.1: Predicción serie temporal demanda

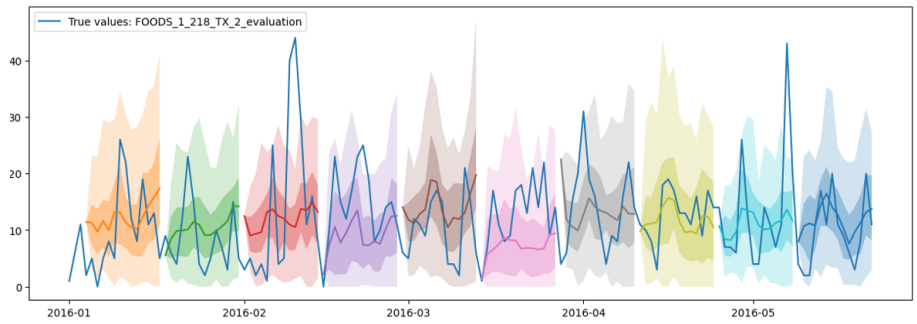

Figura 3.2: Predicción serie temporal demanda (con peor rendimiento)

En definitiva, las series aquí presentadas no dejan de tener una gran variabilidad difícil de comprender para los modelos. Sin embargo, podemos ver cómo captura ciertos patrones, como es el aportar 2 picos de demanda en cada período de 14 días, que podemos intuir que estará relacionado con la estacionalidad semanal que puede tener el consumo de ciertos productos.

3.2. Enfoque Optimización Predicción

En consonancia con lo expuesto en el apartado anterior, vamos a mostrar gráficas del proceso de predicción llevado a cabo en este enfoque. Por introducir contexto, este proceso predice por separado los valores MIN y los valores MAX (tenemos el doble de series y el doble de modelos), en este caso las gráficas mostradas hacen referencia a casos de predicción del valor MAX.

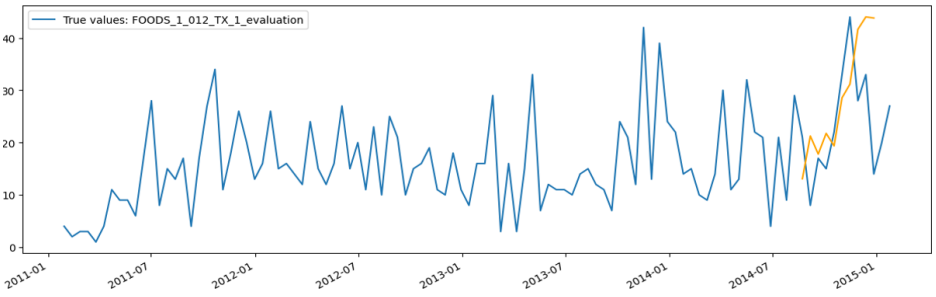

Figura 3.3: Predicción serie temporal MAX

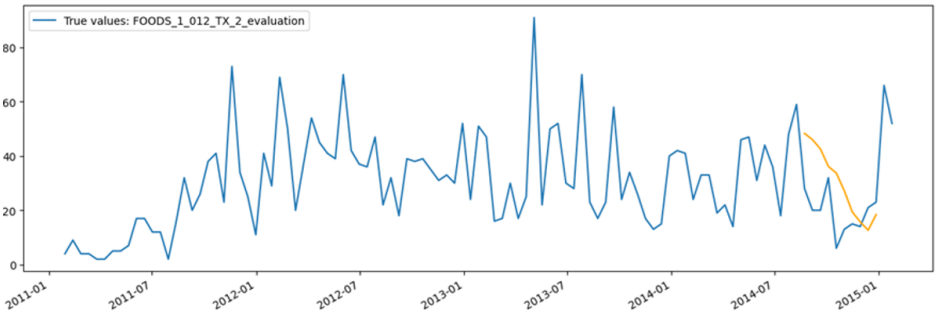

Figura 3.4: Prediccións serie temporal MAX

Tenemos que destacar que el proceso previo de optimización hace que nuestra granularidad sea bisemanal, reduciendo la cantidad de datos de entrenamiento por serie de 1825 datos (5 años de histórico) a un total de 130 datos (aprox.). Esto limita, por un lado, el conjunto de entrenamiento definido para el modelo, como la profundidad del parámetro de contexto que pueden usar ambos modelos.

3.3. Comparación

En este apartado desgranamos los resultados a nivel de costes de los dos enfoques en el período de evaluación que habíamos determinado (01/2016-05/2016). En la figura 3.5 podemos ver la evolución del inventario de un producto a lo largo de este periodo. En verde encontramos los valores de MAX que se actualizan cada 14 días y determinan el umbral de reabastecimiento (la variable *stock* mide el inventario al final del día, por eso vemos que los puntos de la variable no tocan este umbral). En marrón podemos apreciar los valores del umbral MIN, que también se actualizan cada 14 días y determinan el momento del reabastecimiento.

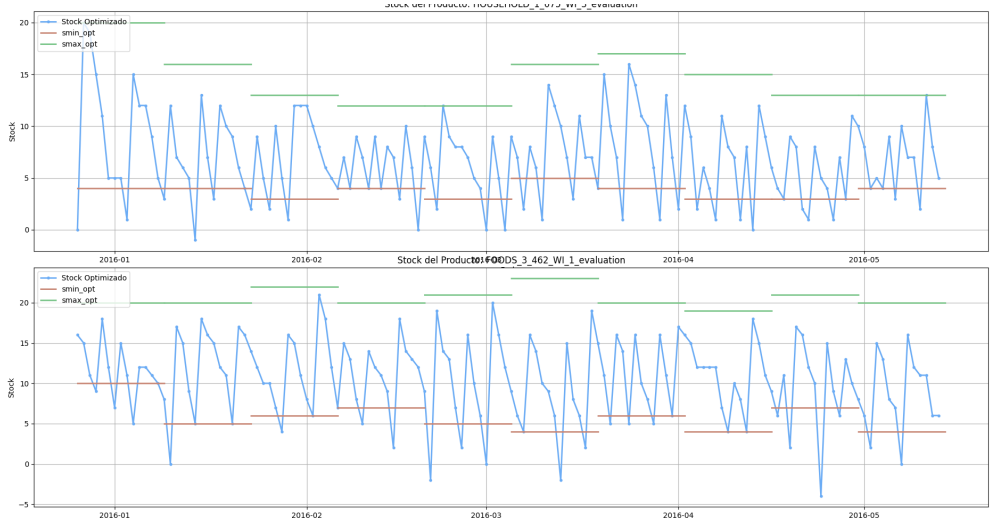

Figura 3.5: Evolución stock periodo evaluación

3.3.1. Coste total

En este primer apartado, vamos a presentar el coste total del periodo de evaluación, que hace referencia a la suma de los 3 costes planteados (envío, ruptura y tirado).

Tipo solución	Coste	Variación porcentual
Modelo Base	2603141	—
Optimización-Predicción	2662048	+2.26 %
Predicción-Optimización	2586016	-0.6 %

Cuadro 3.1: Tabla coste total

La panorámica general del proyecto se ve en la tabla 3.1, donde podemos ver el mejor comportamiento del proceso de Predicción-Optimización (clásico) frente al de Optimización-Predicción. La mejora en términos porcentuales es escasa frente al modelo base planteado, pero vamos a desglosar estos costes en diferentes apartados, ya que el coste de envío ha sido claramente sobre ponderado en el proyecto, acumulando el 97 % del coste total.

Esto es debido a que finalmente se abordó el problema con una simplificación de la gestión máxima de número de pedidos, introduciendo una penalización exponencial. El coste del pedido se ve incrementado un 10 % por cada pedido a mayores que gestiona el supermercado por encima de su límite. En un ejemplo práctico, si suponemos un límite de 2 proveedores al día y la solución seleccionada tiene pedidos de 4 proveedores para un mismo día, a los dos primeros proveedores se le imputa el coste habitual (sin penalización) mientras que al 3º se le aplica un incremento del 10 % y al 4º un 21 % ($1{,}1^2$) y así sucesivamente.

3.3.2. Coste sin envío

Como mencionamos anteriormente, seguiremos el análisis dejando de lado la modelización del coste de envío. Eso se traduce en quedarnos con los otros 2 campos que habíamos planteado en nuestro coste: coste de tirado y coste de ruptura. En la tabla 3.2 podemos ver el desglose de costes.

Tipo solución	**Coste**	**Variación porcentual**
Modelo Base	113449	—
Optimización-Predicción	159931	+41 %
Predicción-Optimización	82897	-27 %

Cuadro 3.2: Tabla coste sin envío

Aislando el gran peso que tenía el coste de envío, las diferencias respecto al modelo base son más acentuadas. En este caso, la mejora del enfoque clásico respecto al modelo base fijado supone un ahorro del 27 %, el cual es considerablemente más grande que el caso anterior (en términos porcentuales). Por el contrario, el enfoque que se quería contrastar de optimización-predicción ha proporcionado unos resultados nefastos, empeorando el coste del modelo base en un 41 %.

3.3.3. Coste ruptura

Por último, podríamos analizar cada coste por separado. Sin embargo, desde la empresa se nos ha trasladado que el coste donde suelen poner el foco este tipo de negocios es en el de ruptura (ya que habitualmente utilizan técnicas comerciales para reducir los costes asociados al tirado: promociones, congelados, reutilizados...). Es por eso, que en este último apartado nos vamos a centrar únicamente en los datos de los costes de ruptura.

Es probable que este coste, a diferencia del coste de envío, esté infravalorado. Esto se debe a que nosotros solo hemos introducido la pérdida de la demanda no satisfecha sin incluir posibles multiplicadores (> 1) que intentaran añadir a la modelización comportamientos de posibles clientes descontentos que dejan de acudir a nuestra tienda cuando ven

que el supermercado no provee la demanda que ellos tienen. En la tabla 3.3 podemos ver el resumen de los costes de ruptura.

Tipo solución	**Coste**	**Variación porcentual**
Modelo Base	96981	—
Optimización-Predicción	115949	+5 %
Predicción-Optimización	63312	-35 %

Cuadro 3.3: Tabla coste ruptura

Podemos observar que la mejora en términos de ruptura de inventario es aún mayor que la del apartado anterior. Esto no quiere decir que todo el ahorro que nuestra solución ofrece se traduce en una mejora en el coste de ruptura, es decir, optimizando la pérdida de demanda de productos en nuestra tienda. Esto nos lleva a ser optimistas en las bondades que puede aportar poner este tipo de soluciones en entornos reales.

Capítulo 4

Conclusiones y mejoras

Cerramos este documento y el trabajo final de máster, mostrando una mirada crítica hacia el trabajo desarrollado. Para ello, presentamos algunas conclusiones obtenidas durante la elaboración del trabajo. Además, consideramos adecuado este último espacio para exponer ideas y mejoras que se han ido desestimando debido a la falta de tiempo para llevarlas a cabo en breve período de prácticas.

4.1. Conclusiones

El objetivo planteado desde la empresa desde un inicio ha supuesto todo un reto en la parte de optimización. Como ha quedado probado, el gestionar una gran cantidad de productos hace aumentar de forma exponencial el número de restricciones del problema de optimización asociado. La exploración de diferentes enfoques ha sido tediosa, pero finalmente el uso de heurísticas ha sido el único camino a seguir. Finalmente, el algoritmo seleccionado para esa parte ha sido un algoritmo genético. A pesar de eso, la restricción de tiempo de cómputo que hemos fijado ha implicado trabajar con poblaciones iniciales muy limitadas en tamaño. Eso nos ha llevado a definir una población inicial cuyos individuos conocíamos que determinaban soluciones con optimalidad razonables, mejorando así el rendimiento de las soluciones finales del algoritmo genético.

A pesar de los problemas computacionales, hemos podido llevar a cabo la comparación de los enfoques que interesaban a la empresa, devolviendo resultados claros y con un trabajo técnico importante que sustentan una conclusión sólida. El enfoque 'clásico' presenta mejor funcionamiento, que nosotros achacamos a la menor dificultad de predicción de las series de demanda respecto a las de MIN o MAX.

La selección de los modelos de Aprendizaje Profundo lo consideramos todo un éxito. Nos ha permitido abordar el problema de predicción de forma rápida y efectiva, reduciendo tiempos de implementación. Esta agilidad a la hora de implementar los modelos ha servido para dedicar una mayor parte del proyecto a la parte de optimización. Podríamos afirmar que del tiempo total de trabajo realizado en el proyecto, un 70 % ha sido dedicado en tareas de Optimización frente al 30 % restante en tareas de Predicción.

Además, la creación de una arquitectura propia ha permitido que podamos aprovechar la fácil implementación de modelos globales a la vez que intentábamos individualizar (de forma muy limitada) la predicción de cada una de nuestras series. En este caso, el conocimiento de los profesionales de la empresa ha marcado la diferencia, ya que nos han permitido ir a lo fundamental y tener una solución técnica en un corto espacio de tiempo, que ha permitido presentar los resultados en un periodo de 3 meses.

Además, esperamos que este experimento sirva como semilla de proyectos reales que

contribuyan a mejorar la eficiencia económica en el sector de los supermercados, gracias a la reducción en el desperdicio de alimentos, lo cual tiene una justificación tanto social como ética.

4.2. Mejoras

En este último apartado intentamos recoger algunas ideas que han surgido en el proceso de trabajo, donde hemos encontrado continuaciones o mejoras del proyecto interesantes pero inabarcables por cuestión de tiempo.

- Tanto DeepAR como TFT nos permiten obtener predicciones probabilísticas (predecir un percentil X). Esta información no ha sido usada en ningún momento, pero sería interesante aprovechar esta información haciendo uso de planteamientos de Optimización Robusta, buscando aportar soluciones más resilientes a posibles cisnes negros.

- En el enfoque de Optimización-Predicción, el proceso de predicción se hace con 2 series temporales distintas por producto. Una de las posibles vías a explorar es plantear una serie temporal bidimensional por producto (MAX,MIN) y utilizar las mismas técnicas para este nuevo tipo de series.

- Uno de los grandes puntos que se discutió en el inicio del proyecto fue si trabajaríamos solo con una tienda. Una de las posibilidades evaluadas fue el añadir diferentes tiendas y economías de escala (reducción de costes en los envíos). Este supuesto complicaría la modelización del problema y por eso se decidió excluirlo de nuestro caso de estudio, pero es algo a explorar en un futuro.

- Uno de los puntos claves en toda la evolución del proyecto es encontrar un compendio de rendimiento económico y rendimiento computacional, para poder abordar el cálculo de MIN-MAX histórico. Esto llevó a limitar el tiempo que le permitíamos usar al algoritmo genético para la búsqueda del óptimo. Una vez determinado que el enfoque clásico presenta mejores resultados, podemos permitir aumentar de forma considerable el tiempo de cómputo permitido para nuestra solución. Uno de los elementos en los que utilizaríamos esa mayor capacidad de cómputo es en la mejora de la variabilidad de la población inicial del algoritmo genético, la cual limita de forma clara la optimalidad de las soluciones encontradas.

- Como trasladábamos en el capítulo de comparación de enfoques, el coste de envío ha sido sobredimensionado en nuestro estudio. Por lo tanto, una de las claves conceptuales a replantear en posibles desarrollos futuros es la conceptualización del envío como característica del negocio.

Bibliografía

[1] Harris, F. W. (1913). *How Many Parts to Make at Once*. The Magazine of Management, 10(2), 135–136.

[2] Makridakis, S.; Spiliotis, E.; Assimakopoulos, V. (2022). *M5 accuracy competition: Results, findings, and conclusions*. International Journal of Forecasting, 38(4), 1346–1364. doi:10.1016/j.ijforecast.2021.11.013 .

[3] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, Christian Gagné. *DEAP: Distributed Evolutionary Algorithms in Python*. Disponible en: https://deap.readthedocs.io/en/master/.

[4] Bryan Lim, Stefan Zohren. *Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting*. Proceedings of the 3rd International Conference on Learning Representations (ICLR), 2021. Disponible en: https://arxiv.org/abs/1912.09363 .

[5] David Salinas, Valentin Flunkert, Jan Gasthaus, Tim Januschowski. *DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks*. International Journal of Forecasting, Vol. 36, Nº 3, 2020, pp. 1181–1191. DOI: https://doi.org/10.1016/j.ijforecast.2019.07.001 .

[6] Alex Alexandrov, Tim Januschowski, Jan Gasthaus, David Salinas, Lorenzo Stella, Matthias Seeger, Thomas Flothow, Syama Rangapuram, Konstantinos Benidis. *GluonTS: Probabilistic Time Series Models in Python*. Amazon Web Services, 2020. Disponible en: https://github.com/awslabs/gluon-ts.