Trabajo Fin de Máster
Máster en Ingeniería Industrial

Sistema de gestión del inventario en un almacén de piezas de repuesto. Análisis de la demanda y determinación del stock de seguridad.

Autor: Eduardo Otaolaurruchi Caro

Tutor: Pedro Luis González Rodríguez

Dpto. Organización Industrial y Gestión de Empresas I
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2023

Trabajo Fin de Máster
Ingeniería Industrial

Sistema de gestión del inventario en un almacén de piezas de repuesto. Análisis de la demanda y determinación del stock de seguridad.

Autor:

Eduardo Otaolaurruchi Caro

Tutor:

Pedro Luis González Rodríguez

Catedrático de Universidad

Dpto. de Organización Industrial y Gestión de Empresas I

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2023

Trabajo Fin de Máster: Sistema de gestión del inventario en un almacén de piezas de repuesto. Análisis de la demanda y determinación del stock de seguridad.

Autor: Eduardo Otaolaurruchi Caro

Tutor: Pedro Luis González Rodríguez

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2023

El Secretario del Tribunal

A mi familia

A mis profesores

Agradecimientos

Dedicar este proyecto de fin de máster a todas las personas que han estado junto a mi a lo largo del camino, familiares y amigos. Sin vosotros nunca habría conseguido llegar al final.

Agradecimiento especial a Eduardo Fuentesal, director ejecutivo de Tier1, S.A., por apoyarme con el proyecto y darme la posibilidad de llevar a cabo este estudio en el contexto de un proyecto importante para la compañía, posibilitándome adecuar conceptos académicos a datos reales.

Finalmente, a mi tutor Pedro Luis Gonzáles, gracias por tus horas de dedicación, por tus consejos y sobre todo por tu confianza en mí.

Resumen

Este trabajo de fin de máster se centra en el análisis y propuesta de soluciones para uno de los problemas críticos en uno de los proyectos más importantes de Tier1 S.A., empresa en la que trabajo. El proyecto en cuestión se enfoca en brindar soporte técnico a los equipos microinformáticos de un cliente, asegurando su operatividad las 24 horas.

Se empleará una simulación codificada en Python para sacar conclusiones que ayuden a optimizar los recursos disponibles, mejorando la calidad del servicio y fomentando la mejora continua de la empresa para elevar su competitividad. El estudio se centra en 25 del almacén donde guardamos los repuestos en Sevilla.

En definitiva, este estudio supone un método efectivo para la gestión de inventarios y la política de aprovisionamiento en la empresa, recomendando niveles de stock para los artículos que minimicen los costes y garanticen el cumplimiento de los acuerdos de servicio con el cliente.

Abstract

This master's thesis focuses on analyzing and proposing solutions for one of the critical problems in one of Tier1 S.A.'s most important projects, where I am currently employed. The project revolves around providing technical support for the client's microcomputer equipment, ensuring their 24/7 operation.

A Python-based simulation will be used to draw conclusions that help optimize available resources, improve service quality, and foster continuous improvement to enhance the company's competitiveness. The study specifically focuses on inventory management in the warehouse in Seville, involving 25 spare parts.

In essence, this study offers an effective method for inventory management and procurement policy in the company, recommending stock levels for items that minimize costs and ensure compliance with service agreements with the client.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

1 INTRODUCCIÓN

Actualmente, me encuentro trabajando en Tier1 S.A. desarrollando labores de control económico de los proyectos de la empresa y apoyando a la gestión corporativa en la toma de decisiones para la mejora continua de la compañía.

En el presente trabajo de fin de máster se analizará uno de los problemas más críticos dentro de uno de los proyectos en curso más importantes para Tier1, identificando y proponiendo soluciones que permitan implementarse en la operativa diaria de la empresa para así mitigar los riesgos a los que nos enfrentamos durante la vida del proyecto. Como miembro de la organización, he tenido la oportunidad de ser testigo directo del transcurso en el tiempo del proyecto desde prácticamente su comienzo y he comprobado de primera mano el gran impacto que tiene la problemática que se pretende analizar y resolver por medio de este trabajo. Para analizar el problema, se propone una simulación codificada a través de Python.

En este apartado se resume quién es Tier1 y posteriormente se hace alusión a la problemática concreta y al proyecto que se pretende abordar para ofrecer una solución admisible a través del estudio realizado a lo largo de este trabajo. A continuación, se desglosarán los objetivos que se persiguen con el trabajo y su alcance. Finalmente, se concluye el capítulo con un breve resumen de cómo se estructurará el contenido del documento.

1.1. Servicios prestados por Tier1

En Tier1 #SOMOSUNO, somos una empresa con una trayectoria de más de 25 años en el mercado, brindando soluciones eficientes e innovadoras a diversas empresas y entidades. Nuestro compromiso con la excelencia nos ha permitido construir relaciones duraderas con una amplia cartera de clientes, quienes confían en nuestros productos, soluciones y equipo técnico.

Nuestra principal área de especialización se centra en la construcción, evolución e implementación de soluciones de software, el despliegue de infraestructuras de tecnología de la información (TI) y la externalización de procesos empresariales. Además, nos dedicamos al mantenimiento y mejora continua de estas soluciones e infraestructuras, ofreciendo sugerencias para optimizar su funcionamiento.

Nuestros servicios están diseñados para hacer que los negocios de nuestros clientes sean más resolutivos y ágiles, brindando un mayor control, eficiencia en los procesos y aportando un valor añadido a sus operaciones.

El objetivo estratégico de Tier1 es mantener una posición competitiva en el mercado a través de productos innovadores y diferenciados de la competencia, de manera sostenible para nuestros clientes, empleados, proveedores y accionistas. La base de nuestros productos de software se construye a partir de los procesos de negocio más ágiles y organizados de nuestros clientes, desarrollados por equipos experimentados en los aspectos técnicos y funcionales, y basados en una lógica eficaz.

Nuestra red de asistencia técnica cuenta con personal propio que ofrece servicios de mantenimiento tanto en la Península como en Canarias y LATAM. Además, estamos capacitados para brindar nuestros servicios en varios idiomas, incluyendo español, inglés, portugués y francés.

Desde junio de 2018, Tier1 cotiza en el Mercado Alternativo Bursátil (MAB) bajo el símbolo TR1, lo que demuestra nuestra solidez y compromiso con la transparencia y la excelencia empresarial.

En Tier1, ofrecemos una amplia gama de soluciones para satisfacer las necesidades de nuestros clientes. Entre ellas destacamos las siguientes:

- Soluciones software personalizadas: Contamos con una amplia experiencia como desarrolladores de soluciones software, lo que nos permite ofrecer a nuestros clientes una cartera de productos ágil y adecuada a sus necesidades. Nuestros productos se destacan por su eficiencia en costos y resultados, y cubren diversas áreas funcionales de la empresa en cualquier sector de actividad privada o pública. Algunas de nuestras soluciones incluyen el ERP Atractor y la plataforma omnicanal de ventas Comerzzia.

- Despliegue y mantenimiento de infraestructuras TI: Nuestros servicios de despliegue y mantenimiento de infraestructuras IT ayudan a nuestros clientes a maximizar la disponibilidad de sus sistemas y servicios. Brindamos soluciones para instalación, renovación y actualización de equipos informáticos y de comunicaciones. Nuestro servicio de microinformática ofrece soporte para el ciclo de vida completo de los equipos informáticos, reparación de equipos y medidas de seguridad cibernética. Además, proporcionamos asistencia en la planificación del entorno de trabajo digital y ofrecemos soluciones para la gestión de periféricos, almacenes y pedidos. En el contexto de estos servicios se sitúa el proyecto de la empresa sujeto a estudio a lo largo de este trabajo. Dicho proyecto consiste en dar un soporte técnico de forma presencial y remota a los equipos microinformáticos de uno de nuestros clientes, ofreciéndoles disponibilidad 24x7 para garantizar la operatividad de sus equipos. Además, cabe resaltar que para este servicio contamos con un almacén de piezas sustitución de aquellas defectuosas en los equipos del cliente. Más adelante, en el capítulo 2 se explicará más detalladamente en qué consiste.

1.2. Justificación del proyecto

La realización de este proyecto se justifica por varias razones fundamentales que abordan los desafíos específicos y las oportunidades de mejora identificadas en el contexto de los servicios de mantenimiento que nuestra empresa presta al cliente. A continuación, se presentan los argumentos que respaldan y justifican la importancia de esta investigación:

1. Optimización de recursos.

La optimización de los niveles de inventario de las piezas utilizadas en el servicio de mantenimiento de las balanzas nos permite reducir los costos asociados al almacenamiento de piezas. Aunque el dinero que tiene mi empresa es limitado, poseemos capacidad "infinita" para almacenar piezas de repuestos, es decir, no tenemos restricciones de capacidad en nuestro almacén. No obstante, mantener un alto nivel de inventario en el almacén supone para la compañía un coste de oportunidad en cuanto a dinero que podría invertir en otros proyectos en lugar de tener "parado" en un almacén.

2. Mejora en la calidad del servicio:

Al establecer niveles de inventario más precisos y ajustados a las necesidades reales del proyecto, podremos garantizar una disponibilidad óptima de las piezas requeridas para el mantenimiento de las balanzas. Esto permitirá brindar un servicio más eficiente y confiable, al evitar retrasos o interrupciones debido a la falta de stock. La mejora en la calidad del servicio fortalecerá la satisfacción del cliente y contribuirá a mantener una relación comercial sólida y duradera.

3. Potencial de mejora continua y competitividad:

Este proyecto sienta las bases para establecer un sistema de gestión de inventario más sólido y eficiente en el ámbito del proyecto más importante de la compañía. La recopilación, monitorización y análisis de datos permitirá identificar patrones de consumo y tendencias de la demanda de los artículos. Además, se podrá comprobar en el futuro si los resultados del trabajo han contribuido a alcanzar los objetivos que se persiguen constituyendo una base sólida para la implementación de estrategias de mejora continua. Al perfeccionar nuestra capacidad para gestionar el inventario de manera óptima, fortaleceremos nuestra competitividad en el mercado, consolidando nuestra posición como proveedores confiables y eficientes.

En conclusión, la justificación de este proyecto radica en la necesidad de optimizar los niveles de inventario de las piezas utilizadas en el servicio de mantenimiento, con el objetivo de optimizar recursos, reducir costos, mejorar la calidad del servicio, optimizar la planificación y potenciar la mejora continua y la competitividad. Estas razones respaldan relevancia de esta investigación, brindando un marco sólido para el logro de mejoras significativas en la gestión de inventario y la eficiencia operativa de nuestra empresa.

1.3. Objetivos y alcance

El objetivo principal será establecer un método que permita dar una solución efectiva al sistema de gestión de inventarios en la empresa.

Para alcanzar el objetivo principal, se perseguirán una serie de objetivos específicos:

- Establecer un stock de seguridad adecuado para los artículos del almacén. Este método debe proporcionar una solución que procure que no se produzcan roturas de stock que supondrían incumplir los tiempos de resolución establecidos por los SLA e incurrir en penalizaciones sobre el recurrente además de disminuir la satisfacción del cliente.

- Proporcionar un stock máximo óptimo. Como ya se comentaba, excederse en el stock necesario para servir la demanda de artículos tiene un coste de oportunidad para la empresa. Se intentará determinar, en la medida de lo posible, el valor del stock máximo que minimice el coste de oportunidad por almacenar el material.

- Sugerir e implementar cambios en la política de aprovisionamiento de la empresa.

En cuanto al alcance de este trabajo, se pretende analizar en profundidad aquellos artículos que se consideran más críticos en la gestión del proyecto, esto es, aquellos artículos que tienen un alto nivel de consumo, un coste elevado o que en el histórico del proyecto han sufrido roturas de stock. Por tanto, no se estudiará el almacén en su totalidad, al menos en este trabajo. Además, el análisis sobre inventarios se realizará únicamente en el almacén de Sevilla sin considerar inventarios adicionales de los técnicos, llamados Van Kits que se explicarán más adelante.

1.4. Estructura del documento

Una vez introducido el estudio con los objetivos generales que se pretenden con el proyecto y el alcance de este, los siguientes capítulos servirán para detallar la metodología llevada a cabo en el trabajo y los resultados que se obtienen.

El capítulo dos servirá para poner en contexto el proyecto que engloba este estudio explicando en qué estado se encuentra en la actualidad.

En el capítulo tres se explica detalladamente el caso de estudio con la metodología aplicada para llegar a los resultados que se van a analizar. En este apartado, se detallan los datos necesarios para el estudio y su procedencia y las hipótesis que se han tenido en cuenta a la hora de aplicar la metodología.

En el cuarto capítulo se muestra la implementación en Python explicando cada parte del código haciendo referencia al paso de la metodología con el que se corresponde.

En el capítulo quinto se analizan los resultados obtenidos de los artículos considerados en el estudio comentando las acciones a implementar para adaptar el sistema de gestión de inventario actual de la empresa a las oportunidades identificadas en este estudio. En un primer lugar, se muestran los resultados de la primera fase de la metodología que sirven para identificar los niveles de stock mínimo óptimos. Finalmente, se comentará la segunda fase que permite hacer un pequeño análisis de sensibilidad a la política de aprovisionamiento de los diferentes artículos del estudio.

El sexto y último capítulo sirve para cerrar el proyecto realizando algunas conclusiones generales de los resultados que se han obtenido, comentar los siguientes pasos a llevar a cabo en la empresa y resaltar posibles estudios futuros que puedan complementar y mejorar este proyecto.

2 ESTADO ACTUAL DEL PROCESO

Como ya se comentaba en el capítulo introductorio, la venta de software ha sido el gran activo de la empresa en los últimos años. No obstante, la línea de negocio dedicada a operaciones está creciendo mucho en Tier1 y en lo que va de año supera en ejecución a los proyectos de software a pesar de que el margen de la línea de software es mucho más alto. En este capítulo, se pretende poner el contexto el proyecto para el que se realiza este estudio.

El proyecto para el que se propone este estudio pertenece a la línea de operaciones y consiste en dar un servicio de soporte presencial y remoto de mantenimiento al parque completo de balanzas, para el pesaje de los alimentos, de todos los supermercados que tiene en la península nuestro cliente. Para respetar al máximo la confidencialidad del cliente no se va a revelar ningún dato acerca de su identidad.

El servicio de mantenimiento que la empresa ofrece a nuestro cliente está enfocado en un soporte integral que tiene como fin solventar cualquier tipo de incidencia que presenten las balanzas de los supermercados. Nuestro compromiso como proveedores es garantizar que todo el parque de balanzas instalado en la península esté operativo durante el horario de apertura de los supermercados. Para hacer una idea de la magnitud del proyecto, el parque actual de balanzas inventariadas en los supermercados del cliente, distribuidos en distintos puntos de la península, se sitúa por encima de las 6.500 balanzas repartidas a lo largo de más de 500 centros.

Clasificamos las incidencias según la naturaleza de su causa origen en dos tipos:

- Incidencias de software: este tipo de problemas se deben a fallos del sistema operativo de las balanzas; las balanzas son auténticos ordenadores y poseen tecnología PC de alta gama amplia memoria RAM, potente procesador Intel y sistema operativo (habitualmente Windows).

  Las incidencias más típicas se producen por lentitud de las balanzas, problemas de comunicación con los servidores, desconfiguraciones o fallos en la actualización de los precios de los alimentos. Estas incidencias se resuelven en la inmensa mayoría de las ocasiones desde nuestro centro de servicio de soporte remoto, conocido como nivel 1. En un pequeño porcentaje de estas incidencias se requiere la intervención in situ de un técnico de campo para solventar los problemas tipo software que presentan las balanzas.

- Incidencias de hardware: estos problemas están ligados al mal funcionamiento de cualquier parte material que compone una balanza. Las balanzas se componen de pequeñas piezas o artículos que integran todo su hardware.

Algunos de estos componentes se explican a continuación a partir de una imagen de una de las balanzas de las que solucionamos componentes.

Ilustración 1 : Balanza XC II 800 Pro de Bizerba

Se trata de la balanza XC II 800 Pro del fabricante Bizerba entre los cuales destacamos los siguientes componentes:

1. Pantalla táctil para la marcar el tipo de alimento que se está pesando. Integra además el sistema operativo de la balanza a través de otros artículos como son la placa base y las tarjetas SD para la memoria.

2. Zona de pesaje. Sobre la superficie horizontal gris se depositan los alimentos para su pesado. El artículo que realiza la función de pesado se conoce como célula de pesaje.

3. Zona de impresión. La parte inferior de la balanza se conoce como carro de impresión si se hace referencia al conjunto en su totalidad. Dicho carro de impresión se compone de otros artículos como son la impresora, el cabezal de impresión (cuya función es sellar el contenido del ticket) o la etiquetadora, integrando a su vez internamente un complejo sistema de engranajes, correas, bisagras, muelles y tapas.

Todas estas piezas son susceptibles de averías y para nuestra empresa garantizar la operatividad de las balanzas cuando surgen estos problemas supone un reto mucho mayor que cuando se trata de una incidencia de software. Para resolver este tipo de incidencias contamos con un equipo de técnicos de campo repartidos por toda la península dando cobertura a todos los supermercados del cliente.

En la empresa disponemos de un almacén principal ubicado en Sevilla donde se almacenan los repuestos de las piezas que son enviados a los técnicos cuando un supermercado de la zona a la que da cobertura le falla algún componente de sus balanzas.

Ilustración 2: Envíos desde el almacén de Sevilla al resto de la península

Como se puede observar en la imagen, nuestros técnicos de campo están distribuidos por distintos puntos de la península. Los destinos a los que más se suele mandar el material están señalados con flechas de color rojo, aquellos con envíos de material medio en amarillo y los que se indican en azul son los envíos menos frecuentados. La escasez de envíos en los puntos azules se debe a que el técnico de la zona estuvo poco tiempo trabajando para la empresa y ahora damos el servicio en dicha zona a través de otro técnico que ha ampliado su zona de intervención como es el caso de Badajoz a la que se atiende con técnicos que utilizan material de Sevilla. También se puede deber a que la zona se ha cubierto recientemente con un técnico que acaba de incorporarse al equipo o que ha sido reubicado de zona, como es el caso de Granada.

Actualmente, contamos con presencia de técnicos en las siguientes zonas:

Tabla 1: Técnicos por zonas

ZONA/ PROVINCIA	ZONAS DE COBERTURA DETALLADAS
País Vasco	2 técnicos repartidos por la comunidad autónoma
Comunidad Valenciana	3 técnicos repartidos por la comunidad autónoma
Comunidad de Madrid	4 técnicos repartidos por la comunidad autónoma
Cataluña	3 técnicos repartidos por Barcelona, 1 técnico en Gerona y 1 técnico en Tarragona
Baleares	1 técnico en Mallorca
Islas Canarias*	1 técnico en Lanzarote, 1 técnico en Fuerteventura, 5 técnicos en Tenerife y 5 técnicos en Gran Canaria
Zamora	1 técnico encargado de la provincia y alrededores
Murcia	1 técnico encargado de la provincia
Andalucía	7 técnicos en Sevilla*, 1 técnico en Granada, 1 técnico en Almería, 2 técnicos en Málaga, 1 técnico en Córdoba y 2 técnicos en Cádiz.

En total contamos con 43 técnicos repartidos por toda la península. No obstante, cabe resaltar que los 12 de las Islas Canarias están bastante más implicados en otro proyecto de la compañía diferente al de mantenimiento de balanzas. Además, hay que considerar que tres de los técnicos de Sevilla son los encargados del almacén y sólo algunos de ellos repara balanzas en el mismo almacén. Por tanto, el total de técnicos repartidos por la península para atender las incidencias del proyecto de balanzas se sitúa en torno a los 35 aproximadamente.

De forma complementaria, cada técnico dispone de un pequeño kit con los repuestos que más suelen emplear para la reparación de las balanzas conocido como Van Kit. El término traducido del inglés se refiere al kit piezas que lleva el técnico en su furgoneta. Esta operativa permite reducir el número de envíos de material a los técnicos minimizando los costes logísticos del trasporte de material. Esto es posible dado que se suelen agrupar los envíos de material mandando distintas referencias al técnico en cuestión cuando los inventarios de sus Van Kit se van reduciendo. Además, el hecho de tener un pequeño inventario de artículos garantiza que si se tiene el repuesto necesario para resolver una determinada incidencia el tiempo de respuesta es más rápido ya que nos ahorramos tiempos de espera mientras el material llega y, por tanto, mejoramos la calidad del servicio que proporcionamos logrando una mayor satisfacción de nuestro cliente.

Contextualizado el proyecto, se puede comprender la dificultad de gestionar un proyecto de semejante magnitud. Uno de los retos más desafiantes a los que nos hemos enfrentado las personas que participamos en el proyecto da nivel organizativo ha sido la gestión de inventarios.

Antes de comenzar este proyecto, el servicio de mantenimiento de las balanzas era prestado por los mismos fabricantes de las balanzas: Bizerba, Dibal y Epelsa. Sin embargo, el cliente tomó la decisión de convocar un concurso público con el objetivo de buscar una propuesta más favorable en términos económicos.

Durante el proceso de evaluación de las propuestas presentadas, nuestra empresa logró destacarse por encima de los fabricantes de las balanzas mediante una oferta económica más competitiva. Esta circunstancia resulta significativa, ya que la elección de nuestra propuesta demuestra que nuestro enfoque y estrategia fueron considerados superiores por parte del cliente.

La adjudicación de este contrato representa un hito importante para nuestra empresa, ya que no solo nos consolida como proveedores de servicios de mantenimiento, sino que también demuestra nuestra capacidad para competir exitosamente con los fabricantes de los equipos en su propio campo. Esta experiencia nos ha permitido fortalecer nuestra reputación en el mercado y nos posiciona como una opción confiable y eficiente en el ámbito del mantenimiento de balanzas en la actualidad.

No obstante, es cierto que al comienzo del proyecto nuestra experiencia en la reparación de balanzas no era la misma que tenían los mismos fabricantes. A pesar de nuestra demostrable experiencia en el mantenimiento de pasillos de supermercados, el proyecto de mantenimiento de balanzas suponía un reto nuevo para la compañía. Durante la fase inicial de puesta en marcha fuimos acompañados por los fabricantes mientras absorbíamos poco a poco la totalidad del servicio. Finalizada la fase de transición, se nos proporcionó un inventario inicial de piezas que, lamentablemente, no se ajustó de forma adecuada a las necesidades específicas de nuestro proyecto. Basándonos en el asesoramiento de los fabricantes, decidimos adquirir 10 unidades de cada artículo que nos recomendaron.

Sin embargo, a medida que transcurrió el tiempo, nos dimos cuenta de que este inventario no era el más adecuado para cubrir las incidencias que iban surgiendo. Descubrimos que muchos de los artículos que habíamos comprado apenas se consumían, lo que resultaba en un exceso de stock en el almacén. Al mismo tiempo, nos enfrentamos a la situación opuesta, donde había otros artículos que se consumían a un ritmo mucho más alto de lo anticipado, lo cual nos generó escasez de existencias al solo disponer de 10 unidades.

Esta disparidad entre la demanda real y el inventario existente nos planteó desafíos significativos en términos de eficiencia y gestión de recursos. Nos dimos cuenta de que era necesario realizar un ajuste en nuestro enfoque de gestión de inventario para garantizar un suministro óptimo de piezas, evitando tanto el exceso como la escasez.

Como respuesta a esta situación, se llevó a cabo un análisis más detallado de la demanda y el consumo de cada artículo en particular. A través de esta evaluación, hemos identificado los artículos que se consumen a un ritmo más acelerado y hemos ajustado nuestros niveles de inventario para asegurar una disponibilidad adecuada.

Durante los últimos meses del proyecto, el punto de pedido en la empresa se fijó en el momento en que el stock de un artículo baja del stock mínimo. Este stock mínimo fue establecido tras analizar el consumo histórico de cada uno de los artículos a los 8 meses desde el arranque del proyecto. El criterio para establecer el stock mínimo fue proponer una cantidad relacionada con los meses de inventario que se quiere tener en stock. Este criterio se puede entender mejor formulado:

$$Consumo\ medio\ al\ mes_{artículo} = {Consumo\ histórico_{artículo}}/{8}$$

La ecuación de arriba nos permite obtener el promedio mensual de consumos de un artículo. De esta forma, se estableció como criterio general que mínimamente se deberían tener entre 1,5 y 2 meses de stock mínimo para los artículos que habían tenido una rotación considerable en la vida del proyecto. Para aquellos artículos de menos consumo se estableció un mes de stock, aunque en la práctica no se llegó a hacer un control del stock mínimo tan exhaustivo como para los artículos de alta rotación. De esta forma, el criterio para establecer un stock mínimo de los artículos fue el siguiente:

$$Stock\ mínimo_{artículo} = Consumo\ medio\ al\ mes_{artículo} * Meses\ de\ Stock$$

$$siendo\ Meses\ de\ Stock = \{1, 1'5, 2\}\ según\ la\ rotación\ del\ artículo$$

Además, se estableció un sistema de seguimiento y monitoreo más riguroso para evaluar constantemente el rendimiento de nuestro inventario y realizar ajustes periódicos en función de las necesidades reales del proyecto. Este enfoque nos ha permitido minimizar los problemas de escasez y exceso de inventario, optimizando así la eficiencia operativa y reduciendo los costos asociados.

El enfoque de este proyecto de fin de máster se basa en proporcionar una solución óptima a los niveles de inventario de los artículos que tenemos en el almacén de Sevilla procurando que, por un lado, no se produzcan roturas de stock estableciendo un stock de seguridad que garantice la disponibilidad de los artículos para la reparación en tiempo de las incidencias y, por otro lado, evitemos tener stock parado en almacén por la falta de demanda fijando un stock máximo.

3 METODOLOGÍA

En este apartado, se va a concretar la metodología empleada para abordar el problema objeto de estudio. Además, a lo largo del capítulo se comenta la procedencia de los datos que se explotan en el estudio y se justifican las hipótesis consideradas en la metodología implementada.

Como ya se comentaba en el capítulo anterior este estudio pretende encontrar una forma adecuada de gestionar el inventario de nuestro almacén de Sevilla: determinar stocks mínimos y máximos, y definir una política de aprovisionamiento correcta.

Para tal fin, se llevará a cabo una simulación que permita ajustar los stocks a valores óptimos en términos económicos y nos permita analizar la política de aprovisionamiento implementada en la simulación. La solución propuesta consiste en evaluar la evolución de las funciones de costes que modelizan el problema para distintos escenarios de demanda estimada de los artículos[1] en un determinado horizonte temporal. Así pues, para cada artículo del estudio se iterarán distintos valores de stock mínimo obteniendo una gráfica de costes que nos proporcione el stock mínimo (stock de seguridad) que minimice la función de costes. En el estudio se dispone de un histórico de datos de consumo de los últimos 9 meses y el horizonte de planificación simulado será el de los próximos 9 meses.

A continuación, se presenta cual es la metodología empleada para alcanzar lo que se pretende y bajo qué premisas se ha modelado la función objetivo del estudio. Para abordar el problema, se propone una metodología en dos fases.

3.1. Metodología. Fase 1

En este capítulo se detalla la fase primera de la metodología implementada en la rutina de Python que nos permiten plasmar los costes para los distintos niveles de stock mínimo simulados de cada uno de los artículos. A partir de las funciones de coste que se representarán desde la propia rutina, se analizará el punto óptimo de stock mínimo de cada uno de los artículos considerados en el estudio con el objetivo de tomar acciones e implementar una determinada política de aprovisionamiento.

A continuación, se propone mediante un esquema los pasos seguidos, que serán explicados en distintos subapartados distribuidos a lo largo del capítulo:

[1] La demanda estimada equivale al consumo previsto de un determinado artículo.

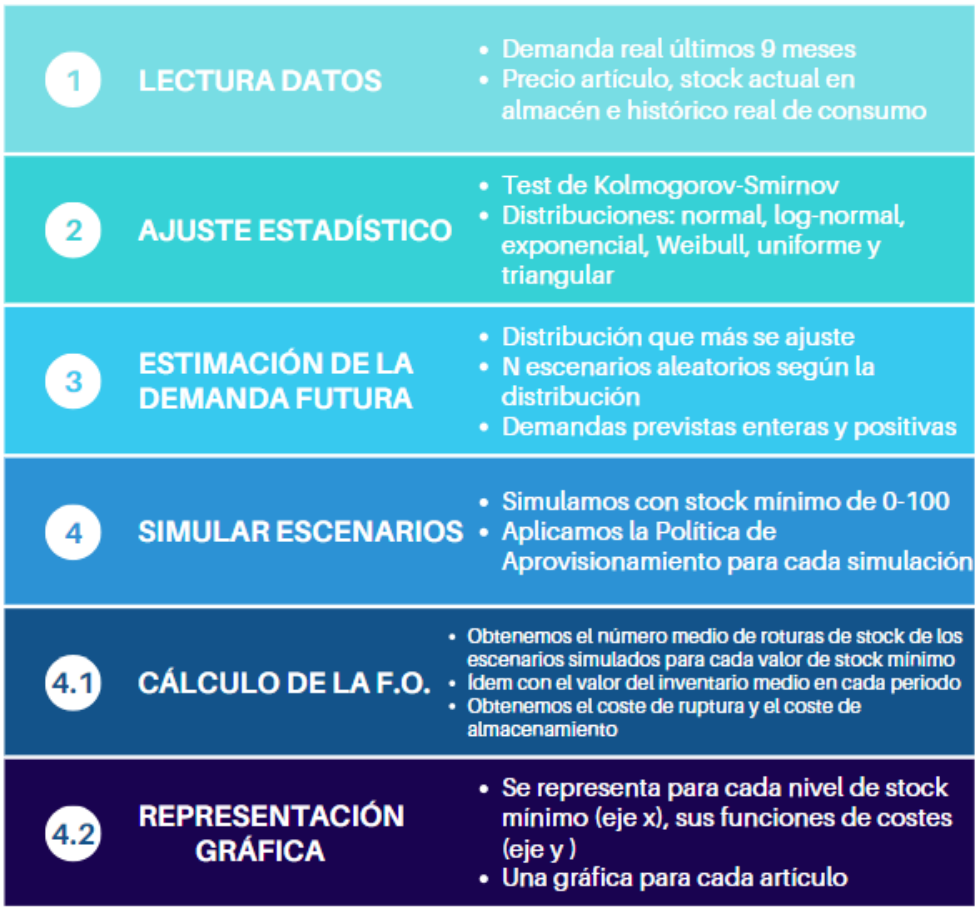


Ilustración 3: Pasos seguidos en la metodología (Fase 1)

Antes de explicar en qué consiste cada paso de la metodología, se darán a conocer los artículos sometidos a estudio. En el almacén de Sevilla contamos con más de 250 artículos inventariados actualmente. Sin embargo, en este proyecto se estudiará un pequeño porcentaje de estos dado que el análisis resultaría interminable. Se detalla, a continuación, el listado de artículos sujetos al estudio:

Tabla 2: Artículos del estudio (I)

Código Artículo	Descripción Artículo	Consumo últimos 9 meses	Coste Artículo (€)	Consumo últimos 9 meses (€)
B61425000184	LPB SCII CPU RETAIL-SC-II	76	619,40 €	47.074,40 €
B61431051020	IMPRESORA XC	42	519,70 €	21.827,40 €
B61421060003	LPB SCII F.A.	67	369,00 €	24.723,00 €
B71058420000	CABEZAL TERMICO SC	124	178,13 €	22.088,12 €
DBCES0126F	MONTAJE ETIQUETADORA	73	236,10 €	17.235,30 €
DBH-KF2002GR	CABEZAL IMPRESIÓN	227	75,85 €	17.217,27 €
E000598002339	CABEZAL ETIQ. PT562 TPH 60MM	126	125,58 €	15.822,58 €
B61421040701	TECLADO SC II N	46	322,41 €	14.830,95 €
B61340227100	WS18 C6/06 C3MI/06 ADW501	31	353,40 €	10.955,40 €

Tabla 3: Artículos del estudio (II)

Código Artículo	Descripción Artículo	Consumo últimos 9 meses	Coste Artículo (€)	Consumo últimos 9 meses (€)
B61425105154	DISPLAY SOLOMON AMBER V1.54	45	243,00 €	10.935,00 €
E000598002331	ETIQUETADORA DE 60 MM 1061	44	207,41 €	9.125,95 €
E00019016760	PLACA PC PICO ITX (ACTUALIZADA)	22	383,78 €	8.443,07 €
D4501003792R	CPU L8/L5 T O ETIQ. BOX/SCA 1UART8X	21	355,15 €	7.458,15 €
B61421602001	IMPRESORA SCII-BCII 100	18	336,00 €	6.048,00 €
B61425020174	LPB SC II HW 7" CPU 1.74	7	760,50 €	5.323,50 €
B61431630021	PARTE SUP CARCASA CPL IMPRES LINEAS 2	19	269,33 €	5.117,27 €
E000119252132	FAC GAMA ALTA VENTA	27	152,38 €	4.114,37 €
B71058410000	CABEZAL IMPRESORA TERMICA 2003-CF11B	24	168,15 €	4.035,60 €
B61431620020	ESTRUCTURA DE BASE IMPRESORA XC 720	21	179,89 €	3.777,69 €
DBCES0146C	MONTAJE ETIQUETADORA CS1100/1200	11	326,19 €	3.588,09 €
B61431079002	PANTALLA TÁCTIL BIZERBA XC	2	1.784,20 €	3.568,40 €
E000119309800	P.G.GA ETHERNET NF	13	272,00 €	3.536,00 €
D4501003732R	CPU L8/L5 ETHERNET + 2 UARTS 8X	9	355,15 €	3.196,33 €
E000598002351	KIT PRT CABEZAL TPH-791	22	139,68 €	3.072,96 €
B61431610020	PUERTA SIN REBOBINADOR XC KPL	22	133,95 €	2.946,90 €

En total 25 artículos que se someterán al estudio. La selección se puede justificar en cuanto al consumo en € de cada artículo del almacén de Sevilla. Se han seleccionado los 25 artículos que más se han consumido atendiendo al coste que supone su consumo. El consumo en € se obtiene a partir de la siguiente ecuación:

$$Consumo\ (€) = Consumo\ (uds) * Coste\ artículo\ (\frac{€}{ud})$$

De esta forma, aunque nos hayamos dejado más de un 90% de los artículos del almacén fuera del estudio, en términos económicos, este estudio contempla casi un 80% del consumo en € a lo largo de los últimos 9 meses. Además, podemos justificar la relevancia de estos artículos atendiendo al histórico de roturas de stock producidas en el último año, ya que la mitad de estos artículos no se han tenido en inventario en algún instante del proyecto en el que fueron requeridos.

Cabe resaltar, que algunos artículos que se ha consumido en muchas unidades a lo largo de los últimos 9 meses se han descartado por el criterio de analizar aquellos que más impacto económico tienen, aunque para futuras ampliaciones de este estudio serían artículos relevantes para su análisis.

3.1.1 Lectura de datos

A partir de ahora, las descripciones de los artículos no se van a indicar, identificando únicamente cada artículo por su código único.

Los datos que se leen desde la rutina de Python para realizar el estudio son los siguientes:

Tabla 4: Histórico de consumo de los artículos del estudio

Código Artículo	oct	nov	dic	ene	feb	mar	abr	may	jun
B61425000184	18	11	14	6	2	3	8	4	10
B61431051020	8	10	14	2	1	4	0	3	0
B61421060003	20	13	11	5	1	4	3	3	7
B71058420000	28	15	21	14	12	12	5	8	9
DBCES0126F	24	6	14	8	3	2	5	8	3
DBH-KF2002GR	40	30	39	15	22	26	23	20	12
E000598002339	30	21	26	12	8	9	5	7	8
B61421040701	9	8	3	10	2	5	3	3	3
B61340227100	11	4	0	2	2	2	1	5	4
B61425105154	11	5	7	1	4	3	4	5	5
E000598002331	14	7	12	5	2	1	1	1	1
E00019016760	4	6	5	1	0	1	1	3	1
D4501003792R	12	0	3	0	1	1	3	1	0
B61421602001	5	1	2	3	0	3	2	1	1
B61425020174	0	3	2	0	0	1	0	1	0
B61431630021	6	0	1	1	3	1	1	5	1
E000119252132	3	8	2	1	3	4	3	3	0
B71058410000	3	4	2	4	4	2	2	2	1
B61431620020	6	0	1	3	3	2	3	2	1
DBCES0146C	7	0	2	0	0	0	0	1	1
B61431079002	1	1	0	0	0	0	0	0	0
E000119309800	2	1	1	0	1	2	2	3	1
D4501003732R	1	0	3	3	1	1	0	0	0
E000598002351	0	5	2	0	8	3	1	1	2
B61431610020	7	0	0	2	3	3	1	2	4

La tabla 4 muestra el consumo real mensual de los últimos 9 meses de los artículos que se van a analizar. Estos datos se leerán desde una de las funciones implementadas Python y están guardados en una hoja de Excel.

Tabla 5: Datos sobre los artículos del estudio

Código Artículo	Stock inicial	Consumo Histórico	Precio Artículo
B61425000184	2	76	619,40 €
B61431051020	3	42	519,70 €
B61421060003	12	67	369,00 €
B71058420000	15	124	178,13 €
DBCES0126F	3	73	236,10 €
DBH-KF2002GR	4	227	75,85 €
E000598002339	5	126	125,58 €
B61421040701	1	46	322,41 €
B61340227100	0	31	353,40 €
B61425105154	19	45	243,00 €
E000598002331	7	44	207,41 €
E00019016760	0	22	383,78 €
D4501003792R	15	21	355,15 €
B61421602001	6	18	336,00 €
B61425020174	0	7	760,50 €
B61431630021	7	19	269,33 €
E000119252132	5	27	152,38 €
B71058410000	31	24	168,15 €
B61431620020	5	21	179,89 €
DBCES0146C	0	11	326,19 €
B61431079002	0	2	1.784,20 €
E000119309800	0	13	272,00 €
D4501003732R	0	9	355,15 €
E000598002351	10	22	139,68 €
B61431610020	13	22	133,95 €

En la tabla 5 podemos apreciar el stock actual del de cada artículo en el almacén de Sevilla a partir del cual se inician las simulaciones del horizonte de planificación, así como el consumo total de los últimos 9 meses de cada artículo y su precio de compra a los fabricantes. Al igual que los datos sobre el consumo mensual, se guardarán en el mismo archivo Excel, pero en una hoja distinta. Para su lectura, se implementa otra función en Python.

3.1.2 Ajuste estadístico

Este se ha llevado a cabo mediante la codificación de la prueba de Kolmogórov-Smirnov. Este test es una herramienta estadística utilizada para identificar la distribución estadística que mejor se ajusta a un conjunto de datos de demanda en un estudio determinado. Hay dos variantes principales de esta prueba: el test de una muestra y el test de dos muestras. Ambos test son aplicables únicamente a distribuciones continuas.

En nuestro caso se utilizará el test de una muestra en el que se compara una distribución subyacente F(x) de una muestra aleatoria con la distribución dada G(x). Existen tres opciones para plantear la hipótesis nula y la hipótesis alternativa correspondiente, las cuales se seleccionan mediante el parámetro "alternative" de la función de Python que nos permite realizar la prueba de Kolmogórov-Smirnov:

- Two-sided (ambos lados): La hipótesis nula plantea que las dos distribuciones son idénticas, es decir, $F(x) = G(x)$ para todos los valores de x; mientras que la hipótesis alternativa sugiere que no son idénticas. En nuestro caso, utilizamos esta opción.

- Less (menor): La hipótesis nula establece que $F(x) \geq G(x)$ para todos los valores de x; mientras que la hipótesis alternativa plantea que $F(x) < G(x)$ para al menos un valor de x.

- Greater (mayor): La hipótesis nula indica que $F(x) \leq G(x)$ para todos los valores de x; mientras que la hipótesis alternativa sugiere que $F(x) > G(x)$ para al menos un valor de x.

Las hipótesis alternativas describen las funciones de distribución acumulada (CDF) de las distribuciones subyacentes, no los valores observados. Por ejemplo, si $x1 \sim F$ y $x2 \sim G$, si $F(x) > G(x)$ para todos los valores de x, los valores en x1 tienden a ser menores que los valores en x2.

El test de Kolmogórov-Smirnov se lleva a cabo siguiendo una serie de pasos. En primer lugar, se ajustan los parámetros desconocidos de la familia de distribuciones especificada a los datos proporcionados, utilizando la estimación de máxima verosimilitud. Esto forma la "distribución nula hipotética", que es la distribución a partir de la cual se muestrearon los datos bajo la hipótesis nula. A continuación, se generan muestras aleatorias de esta distribución nula hipotética y se ajustan los parámetros desconocidos a cada muestra. Se calcula el estadístico entre cada muestra y la distribución ajustada a esa muestra. Estos valores del estadístico forman la "distribución nula de Monte Carlo".

A continuación, se calcula el p valor del test, que será el criterio empleado para rechazar la hipótesis nula, comparando el valor del estadístico correspondiente a los datos reales con los valores del estadístico correspondientes a las muestras aleatorias de la distribución nula de Monte Carlo. El p valor es la proporción de valores del estadístico en la distribución nula de Monte Carlo que son igual o más extremos que el valor del estadístico calculado para los datos observados.

Para rechazar la hipótesis nula en cada una de las distribuciones que se realiza la prueba de Kolmogórov-Smirnov, se ha determinado que el p valor debe ser menor que una constante denominada p alfa a la que se le ha dado el valor de 0,05.

Para cada distribución probada, se considerará la distribución que mejor se ajusta a los datos reales de demanda aquella que tenga un mayor p valor.

3.1.3 Estimación de la demanda futura

Conocida la distribución estadística que mejor se ajusta a los datos de consumo real que se conocen de los artículos y conocidos también los parámetros estadísticos correspondientes a dicha distribución, se estiman 10 escenarios aleatorios de demanda futura.

Dado que algún periodo de demanda estimada en alguno de los escenarios simulados puede tener un valor negativo y también ser un número decimal (algo que ocurrirá en la mayoría de las ocasiones), estos escenarios generados son depurados a través una función codificada en Python que redondeará al entero positivo más cercano. En el caso de que el valor de demanda estimado sea nulo, la función lo pondrá en cero para que en las simulaciones no ocurran cosas extrañas que no pueden producirse en la realidad.

3.1.4 Simulación

Este es el paso último para el cálculo de la función objetivo del problema y la representación gráfica de las funciones de costes de cada uno de los artículos.

Partiendo del stock mínimo de cada artículo a principios del mes de julio se va a simular el horizonte de planificación a partir de los escenarios que se han estimado previamente.

Durante la simulación se van calculando el número de roturas de stock y el inventario en almacén en cada periodo del escenario simulado. Para cada escenario se calcula la media de roturas e inventario en los periodos y, luego, se hace la media entre los escenarios. Este paso se codifica a partir de una función auxiliar que se llama dentro de la función donde se realiza la simulación. Los valores medios de rupturas e inventario son los que se utilizan para obtener las funciones de coste de cada artículo para cada valor del stock mínimo simulado en la rutina de Python.

Para llevar a cabo la simulación, se parte de la hipótesis de que en caso de que tengamos que realizar un pedido de material en alguno de los periodos simulados, bien porque se produzca una rotura stock o bien porque el stock baje del stock de seguridad, los fabricantes nos proporcionarán el material al inicio del siguiente periodo.

Por otro lado, en Tier1 se quiere evolucionar hacia un sistema de revisión continua del inventario de los artículos. En este sistema, no definimos una política de aprovisionamiento con una periodicidad fijada para la emisión del pedido. Las fechas dependerán del ritmo al que se vayan consumiendo los artículos. A mayor consumo también habrá una mayor cantidad de pedidos y viceversa. La cantidad a solicitar será siempre la misma, pero estará condicionada por el volumen óptimo del pedido: pediremos cuando el inventario esté por debajo del stock de seguridad y hasta llegar al stock máximo.

La diferencia de unidades entre el stock de seguridad y el stock máximo se denomina lote. Se aprecia mejor a partir de la siguiente ilustración:

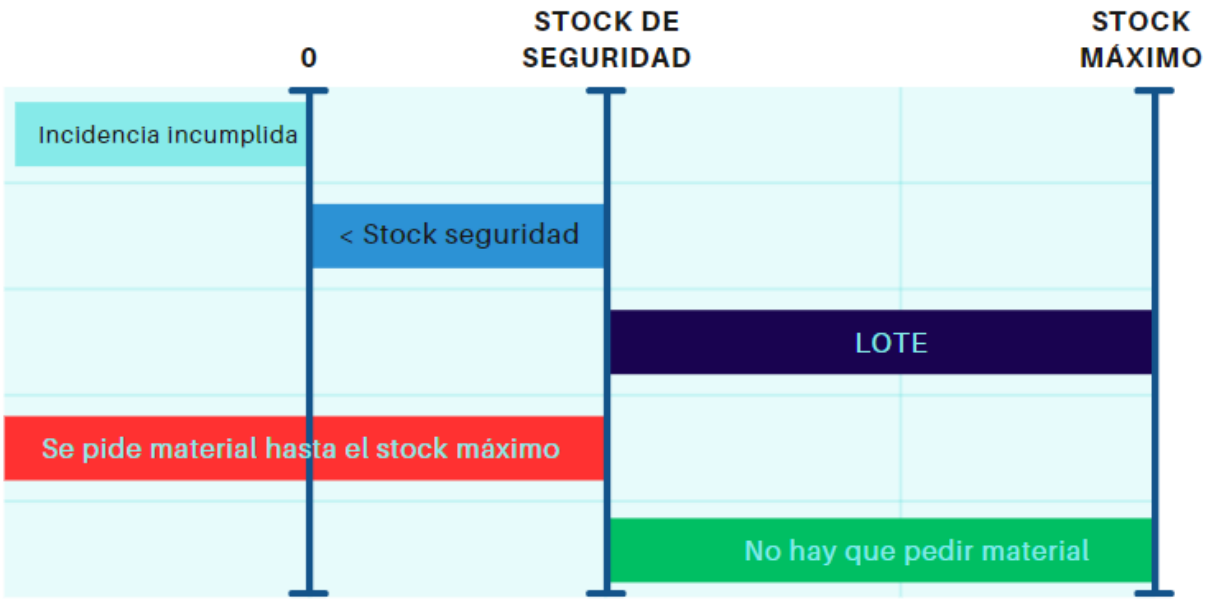


Ilustración 4: Política de aprovisionamiento

Cuando el stock es menor que 0 se produce la rotura de stock y se incumple una incidencia (1 rotura = 1 incidencia incumplida), además hay que pedir material. Cuando el stock está por debajo del stock de seguridad no se produce rotura, pero se tiene que pedir material hasta el stock máximo. Finalmente, si el stock está entre el máximo y el mínimo no hay necesidad de pedir material.

El número de unidades que conforman este lote de pedido se obtiene según el número de meses de inventario que se quieran tener como stock máximo en el almacén. Este número de meses de inventario viene ligado al consumo histórico del artículo. Por ejemplo, si un artículo A se ha consumido 90 veces a lo largo de los 9 meses observados, el consumo medio mensual de A será de 10 unidades. Si parametrizamos el número de meses de inventario este valor nos proporcionará la distancia entre el stock de seguridad y el stock máximo:

$$Lote = Meses\ de\ inventario * Consumo\ medio\ mensual$$

De esta forma, si decidimos que queremos 1 mes de inventario de A, es decir, quiero tener inventario de A para 1 mes, las unidades de artículo que componen el lote serían 10, siendo lote la distancia entre el stock mínimo y el stock máximo.

Para el caso de la empresa, como los artículos que se analizan en este estudio son críticos, se van a tomar 2 meses de inventario para la simulación.

Finalmente, lo único que faltaría por añadir sabiendo cual es la política de aprovisionamiento de la empresa, es que, a partir de los valores medios de roturas e inventarios, se han creado dos funciones dentro de la función de la simulación en las que, por un lado, se calculan las funciones objetivo de costes y, por otro lado, se representan estas gráficamente. Estas funciones de costes se detallan en el siguiente subapartado.

3.2. Función objetivo. Modelado de las funciones de costes

Para evaluar el coste en el horizonte de planificación del inventario, se van a considerar dos funciones de costes para los artículos. Ambas funciones de costes están en €/horizonte temporal siendo el horizonte temporal el número de periodos (meses) que son simulados en la rutina de Python:

- **Coste de almacenamiento.** Representa el coste de oportunidad de la empresa por tener inventario de un artículo en el almacén más el dinero perdido debido a la obsolescencia del material almacenado.

  Para el estudio, se ha modelado el coste de almacenamiento a partir de la siguiente ecuación:

  $$Coste\ \ almacenamiento_{artículo} = Inventario\ medio\ horizonte_{artículo} * T * Precio_{artículo} * Factor$$

  - El inventario medio en el horizonte se obtiene con la siguiente expresión:

    $$Inventario\ medio\ horizonte_{artículo} = \frac{\sum_{t=1}^{T} Inventario\ final_{artículo,t}}{T},$$

    $$siendo\ \ Inventario\ final_{artículo,t}\ el\ inventario\ del\ artículo\ al\ final\ del\ periodo\ t\ y$$

    $$T\ el\ número\ de\ periodos\ considerados\ en\ el\ horizonte\ de\ planificación.$$

  - El precio del artículo es el precio al que compramos a los fabricantes.

  - El factor multiplicativo tiene en consideración el coste de oportunidad de la empresa y el deterioro del material almacenado. Se considerará igual a 0,15 equivalente a un 10% de coste de oportunidad y un 5% por el deterioro de los artículos durante el horizonte temporal.

- **Coste de ruptura.** Hace referencia a la penalización económica que nos aplica el cliente por el incumplimiento de los SLA. La calidad del servicio que prestamos viene ligada al cumplimiento de los SLA. Los SLA (Service Level Agreements) o Acuerdos del Nivel de Servicio son las condiciones de tiempo de resolución acordadas con el cliente para dar el servicio de mantenimiento. En este caso, el cliente nos exige unos SLA muy estrictos en los que nos toleran un porcentaje de incidencias caducadas muy pequeño. Se resumen en la tabla que se muestra a continuación:

Tabla 6: SLA[2]

Tipo de Centro	Severidad[3]	% Cumplimiento SLA
Supermercado tipo 1	1 NBD	95%
	2 NBD	100%
	SV	95%
Supermercado tipo 2	2 NBD	95%
	3 NBD	100%
Supermercado tipo 3	1 NBD	95%
	2 NBD	100%

Como se puede apreciar, según el tipo de centro en el que se produzca la incidencia, se tiene un tiempo (vigencia) para resolver la incidencia. Además, en los supermercados de tipo 1 se pueden dar dos tipos de incidencias las cuales tienen vencimientos de la caducidad[4] diferentes: críticas y no críticas. Se considera que una incidencia es crítica cuando afecta a la sección completa, es decir, la sección de un supermercado ya sea frutería, pescadería, carnicería, etc., está parada y no puede vender. Por su parte, será no crítica si a pesar de darse una incidencia se tienen otras balanzas que mantienen la sección operativa para la venta al público. En los supermercados tipo 2 y 3 no se hace distinción entre la criticidad de la incidencia y se tratan todas las incidencias bajo las mismas condiciones.

Ahora, se explica detalladamente el contenido de la tabla con los SLA:

a) Supermercado tipo 1

- Incidencia crítica: se dispone de 1 NBD para resolver la incidencia antes del vencimiento de su caducidad. El NBD se traduce del inglés "Next Business Day" que quiere decir Siguiente Día Laboral. Esto implica que tenemos hasta el final del siguiente día laborable contando desde el día que nos entra la incidencia para su resolución. Para cumplir con el SLA se requiere resolver el 95% de las incidencias de este tipo. Cuando no somos capaces de cumplir a tiempo con estas incidencias, nos dan 1 día laborable extra: 2 NBD. En este caso el 100% de las incidencias que no somos capaces de resolver en 1 NBD tenemos que resolverlas en 2 NBD para cumplir con el SLA.

- Incidencia no crítica: teóricamente esta severidad significa "Siguiente Visita" lo que implicaría que

[2] No se incluyen los SLA para las incidencias de software ya que en este trabajo nos centraremos únicamente en las incidencias de hardware que se resuelven presencialmente en tienda.
[3] La severidad de las incidencias no son más que una forma de categorizar las incidencias para asignarles el tiempo de vigencia adecuado según el tipo de incidencia que sea. Depende del centro donde se produzca y de la criticidad de la incidencia.
[4] Llamamos vencimiento de caducidad al tiempo de vigencia que tienen las incidencias, entendido como el tiempo que disponemos para resolver la incidencia.

esa balanza no se arreglará hasta que el técnico no realice una visita a la tienda por cualquier motivo. Se llegó a un acuerdo con el cliente para dotar estas incidencias con 7 días de tiempo de vigencia hasta su caducidad. Por tanto, estas incidencias tienen 6 días para ser resueltas más el mismo día en el que entra la incidencia. El porcentaje de cumplimiento requerido es del 95%.

b) Supermercado tipo 2

Todas las incidencias de estos centros se tratan igual sin hacer distinción entre críticas y no críticas como ya se ha comentado. Los SLA exigidos son similares a los del supermercado tipo 1 pero con la diferencia que se cuenta con 2 días laborables en lugar de 1 hasta el vencimiento de la caducidad con un 95% de cumplimiento. De forma análoga, las incidencias no resueltas en este periodo saltan al tercer día laboral en el que tenemos que cerrar el 100% de estas para cumplir con el SLA.

c) Supermercado tipo 3

Este tipo de supermercado tampoco hace distinción entre incidencias críticas y no críticas. Los SLA son idénticos al de las incidencias críticas del supermercado tipo 1.

Estos SLA también impactan sobre la rentabilidad del proyecto. Si incumplimos cualquiera de los SLA en cualquiera de los centros nos aplican una penalización sobre el recurrente que el cliente nos paga mensualmente por los servicios de mantenimiento. Estas penalizaciones se aplican de forma independiente según el tipo de centro, es decir, podemos incumplir en los supermercados tipo 1, pero cumplir en los de tipo 2 y 3; y solamente nos aplicarán penalización por el incumplimiento de los supermercados tipo 1. No obstante, el incumplimiento en cualquier severidad es motivo de aplicación de la penalización.

Para monetizar este coste habrá que identificar el coste que supone cada rotura de stock, para el cual nos basaremos en la siguiente aproximación:

Tabla 7: Penalizaciones según el tipo de supermercado

Penalizaciones por Tipo de Centro				
Tipo	KPIs SLA	N.º Balanzas	Recurrente	Importe
Supermercado Tipo 1	4%	4800	61.200,00 €	2.448,00 €
Supermercado Tipo 2	4%	1200	15.300,00 €	612,00 €
Supermercado Tipo 3	4%	500	6.375,00 €	255,00 €
	Total	**6500**	**82.875,00 €**	**3.315,00 €**

En la tabla 3 se recoge un resumen del recurrente mensual que percibimos por las balanzas que mantenemos. El total de balanzas se ha puesto de manera aproximada con el reparto proporcional real según el tipo de supermercado. El precio que nos pagan por cada balanza mantenida es de 12,75€. Sobre el recurrente total, aplican una penalización del 4% en cada tienda en caso de incumplimiento de los SLA. De esta forma, si se incumplen los SLA de los tres tipos de supermercado nos supone una penalización de 3.315€.

Para el modelo, se considerará que una rotura de stock de un artículo equivale a una incidencia incumplida. No obstante, el incumplimiento de los SLA se produce para un total de incidencias incumplidas mayor que uno ya que el total de incidencias tipo hardware que se cierran al mes se sitúa en torno a las 500.

En la siguiente tabla se muestra el número de caducadas a partir del cual se puede asumir que se incumplen los SLA en cada tipo de supermercado según el histórico de datos de incidencias cerradas al mes:

Tabla 8: Número máximo de caducadas permitidas

Tipo	N.º caducadas
Supermercado Tipo 1	19
Supermercado Tipo 2	7
Supermercado Tipo 3	1

Como la suma total de las incidencias que pueden caducarnos es de 27, modelaremos el coste de ruptura por incidencia mediante una proporción del número de rupturas medias mensuales que se producen en la simulación respecto a las incidencias máximas que pueden caducarnos si la media de rupturas al mes es inferior a 27 y aplicando la penalización total de 3.315€ en caso de que las rupturas medias al mes sean superiores a 27:

$$[1]\ Coste\ \ ruptura_{artículo} = Rupturas\ medias_{artículo} * T * \frac{Penalización\ máxima}{Máximas\ caducadas\ permitidas}$$

$$= Rupturas\ medias_{artículo} * T * \frac{3.315}{27}, \qquad si\ Rupturas\ medias_{artículo} < 27$$

$$[2]\ Coste\ \ ruptura_{artículo} = T * Penalización\ máxima = T * 3.315, \qquad si\ Rupturas\ medias_{artículo} \geq 27$$

La función objetivo que se pretende minimizar es el coste total durante el horizonte de planificación. Esta función se obtiene a partir de la suma de las funciones de costes definidas anteriormente:

$$Coste\ total = Coste\ de\ almacenamiento + Coste\ de\ ruptura$$

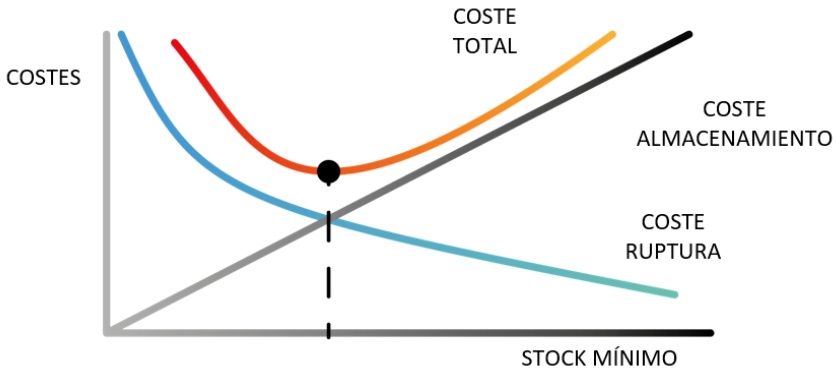


Ilustración 5: Curvas de costes del problema

La ilustración 3 refleja el comportamiento esperado del sistema. Cuando el stock mínimo es cero, el coste de ruptura es máximo y el coste de almacenamiento es mínimo. A medida que se aumentan los niveles de stock mínimo, el coste de ruptura tiende a cero mientras que el coste de almacenamiento aumenta linealmente. La suma de las curvas de coste de almacenamiento y ruptura nos proporciona el valor del coste total. En el mínimo de esta curva se obtiene el stock mínimo que optimiza el problema.

Es importante resaltar que la rutina de Python simula 10 escenarios diferentes de demanda estimada. A partir de estos escenarios se obtienen los inventarios y las rupturas medias en los periodos simulados.

3.3. Metodología. Fase 2

Finalmente, se detalla la segunda fase de la metodología implementada en la rutina de Python en la que a partir de los niveles de stock mínimo obtenidos en la primera fase se iterarán distintos valores de los meses de stock de cada uno de los artículos. Estos meses de stock son los que determinan el lote de pedido o el stock máximo hasta el que se pide. De forma análoga a la fase 1, se representan las funciones de costes para analizar el óptimo de meses de stock de cada uno de los artículos considerados en el estudio con el objetivo de tomar acciones e implementar una determinada política de aprovisionamiento.

A continuación, se propone mediante un esquema los pasos seguidos, que serán explicados en distintos subapartados distribuidos a lo largo del capítulo:

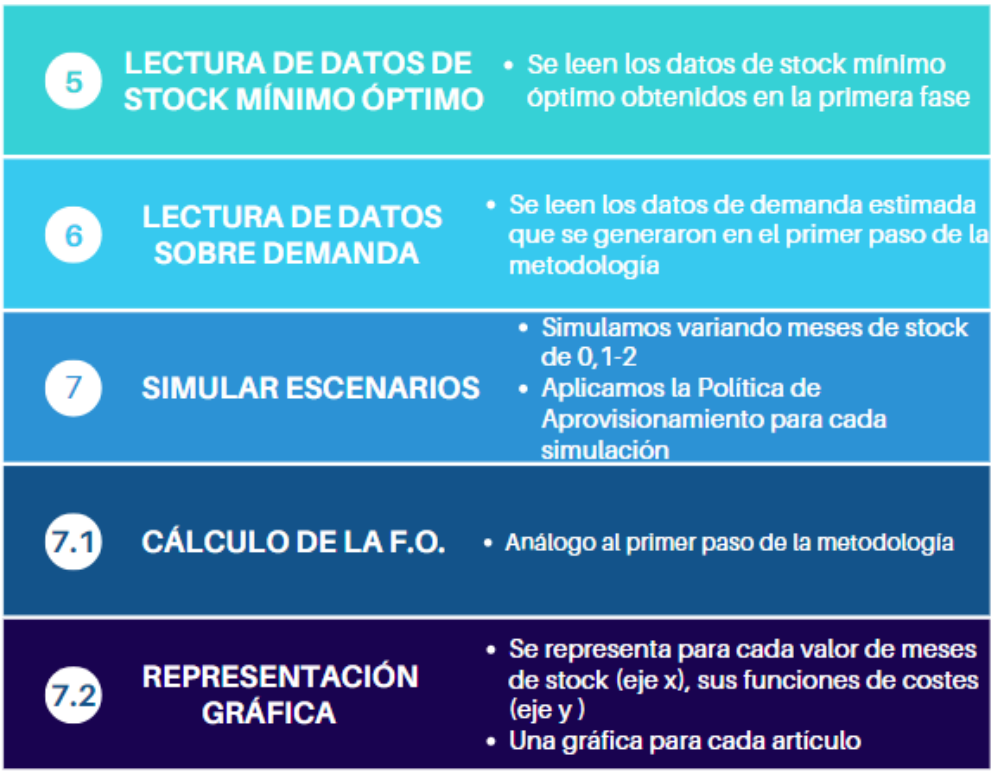


Ilustración 6: Pasos seguidos en la metodología (fase 2)

En esta fase, se leen los resultados de stocks mínimos obtenidos, así como las mismas estimaciones de demanda que se generaron durante la primera fase.

A partir de estos datos, se realizarán las simulaciones de forma análoga a la primera fase, pero las iteraciones se realizan teniendo en cuenta el stock mínimo óptimo de cada artículo y variando los meses de stock que queremos tener en inventario de cada artículo. Mientras se simula, se calculan las mismas funciones de costes explicadas con anterioridad y se representan gráficamente; esta vez en el eje x no se tendrán valores de stock mínimo sino los meses de stock iterados.

4 IMPLEMENTACIÓN EN PYTHON

A continuación, se detalla la codificación en Python de la metodología llevada a cabo en el estudio enlazando cada paso de la metodología a las diferentes funciones implementadas en la rutina.

```
#LECTURA DE DATOS DE EXCEL

def read_file_excel_datos_demanda(filename):

#Leo datos de archivo

    Data={

     'demanda':['A1','J26'],

     }


    excel_document=load_workbook(filename)

    sheet='CONSUMOS'

    datos=Read_Excel_to_NesteDic(filename,sheet,Data['demanda'][0],Data['demanda'][1])

    articulos=list(datos.keys())

    meses=list(datos[articulos[0]].keys())

    demanda={}


    for clave, valor in datos.items():

        demanda[clave] = [valor[mes] for mes in meses]


    return(datos, demanda, articulos, meses)
```

La función read_file_excel_datos_demanda lee los datos la tabla 4 con los datos sobre el histórico de consumo de los artículos. Devolverá una lista donde se condensan los datos, la demanda (consumo) de cada artículo, una lista con los artículos que nos servirán como índices para recorrer los bucles y otra lista con los meses del horizonte de planificación que igualmente nos servirán como índices.

```
def read_file_excel_datos_articulos(filename):
    Data={
      'datos':['A1','D26'],
      }

    excel_document=load_workbook(filename)
    sheet='ARTICULOS'
    datos=Read_Excel_to_NesteDic(filename,sheet,Data['datos'][0],Data['datos'][1])
    stocks = {}
    consumo_hco = {}
    precio_art = {}

    for clave, valor in datos.items():
        stocks[clave] = valor['Stock inicial']
        consumo_hco[clave] = valor['Consumo Histórico']
        precio_art[clave] = valor['Precio Artículo']

    return(datos, stocks, consumo_hco, precio_art)
```

Mediante la función read_file_excel_datos_articulos somos capaces de leer la tabla 5 con los datos de los artículos. A través de la función se obtienen 3 listas con los datos de stock, consumo histórico y precio de los artículos.

Para las dos funciones definidas anteriormente, será necesario el uso de otra rutina de Python que almacenamos en la misma carpeta que nos permitirá hacer la lectura de datos: **from IOfunctionsExcel5 import ***

```
def distribucion_estadistica(demanda,articulos):
    # Lista de distribuciones a probar
    distribuciones = ["norm", "lognorm", "expon", "weibull_min", "uniform", "triang"]
    art_distr={}
```

```
    # Inicialización de variables para almacenar la mejor distribución y sus parámetros
    mejor_distribucion = None
    mejores_params = {}
    demand=[]
    articulos_no_aceptados=[]
    articulos_finales=[]
    for i in articulos:
        #print("Artículo", i)
        demand=demanda[i]
        #print("Demanda", demand)
        mejor_pValor=0
        mejores_params = {}
        aceptaciones=[]
        for distribucion in distribuciones:
            p_alfa=0.05
            #print("Distribución: ", distribucion)
            parametrosAutoAjustados = eval("scipy.stats."+distribucion+".fit(demand)")   #ajusta
los datos de demanda a la distribución
            #print("Parámetros de la distribución", parametrosAutoAjustados)
            estadistico_KS, pValor = scipy.stats.kstest(demand, distribucion,
args=parametrosAutoAjustados)  #Se aplica el test de Kolmogorov-Smirnov
            decision = "SI"
            # Criterio de aceptación
            if pValor > p_alfa:
                #print("pvalor ",pValor)
                decision = "NO"
                aceptaciones.append(0)
```

```
            else:
                aceptaciones.append(1)
            total_acept=sum(aceptaciones)


        # Actualiza la mejor distribución si el pValor es mejor que el anterior
            if pValor > mejor_pValor:
                #print("SI")
                mejor_distribucion = distribucion
                #Recoloco los parametros para que en todas guarde el correcto en su posicion
                mejores_params = {'loc': parametrosAutoAjustados[-2], 'scale':
parametrosAutoAjustados[-1], 'forma': parametrosAutoAjustados[:-2]}
                mejor_pValor = pValor


        #print ("Suma aceptaciones para el artículo", i, "son", total_acept)
        # Se van a descartar del estudio todos aquellos artículos que no se ajusten a ninguna
distribución estadística
        if total_acept == len(distribuciones):
            articulos_no_aceptados.append(i)
        else:
            articulos_finales.append(i)
            #print('Mejor distribución:', mejor_distribucion)
            #print('Parámetros del ajuste:', mejores_params)
            #print('pValor:', mejor_pValor)


            art_distr[i] = {'mejor_dist': mejor_distribucion, 'mejor_params': mejores_params}


    return (art_distr, articulos_no_aceptados, articulos_finales)
```

A partir de la función distribución_estadistica se identifica aquella distribución que mejor se ajusta a los datos de demanda de cada uno de los artículos aplicando el test de Kolmogórov-Smirnov. Además, si ninguna distribución estadística se ajusta lo suficientemente bien a los datos de demanda proporcionados para un determinado artículo, dicho artículo será descartado del estudio. De esta forma, se devuelve una lista con los artículos acompañados de la distribución de mejor ajuste y los parámetros de dicha distribución. También se devuelve el listado de artículos no aceptados, en caso de que no exista ninguna distribución que cumpla el test para algún artículo, y de artículos finales del estudio cuyos datos de demanda si se ajustan a una determinada distribución. Para esta función se hace uso de la librería scipy: **from scipy import stats**

```
def estima_demanda(art_distr, articulos, meses, demanda):
    n_escenarios = 10 # número de escenarios a simular
    n_meses = len(meses)  # número de meses en el próximo año
    estimaciones = {}
    distr_param=art_distr
    for i in articulos:
        #print("Artículo", i)
        estimaciones[i]={}
        for j in range(n_escenarios):
            if distr_param[i]['mejor_dist'] == 'norm':
                mu=distr_param[i]['mejor_params']['loc']
                std=distr_param[i]['mejor_params']['scale']
                estimaciones[i][j]=(np.random.normal(mu, std, size=n_meses))
            elif distr_param[i]['mejor_dist'] == 'lognorm':
                loc = distr_param[i]['mejor_params']['loc']
                scale = distr_param[i]['mejor_params']['scale']
                s = distr_param[i]['mejor_params']['forma']
                estimaciones[i][j]=(lognorm.rvs(s, loc=loc, scale=scale, size=n_meses))
            elif distr_param[i]['mejor_dist'] == 'expon':
                loc = distr_param[i]['mejor_params']['loc']
                scale = distr_param[i]['mejor_params']['scale']
                estimaciones[i][j]=(expon.rvs(loc, scale, n_meses))
```

```
        elif distr_param[i]['mejor_dist'] == 'weibull_min':
            s = distr_param[i]['mejor_params']['forma']
            loc = distr_param[i]['mejor_params']['loc']
            scale = distr_param[i]['mejor_params']['scale']
            estimaciones[i][j]=(weibull_min.rvs(s, loc=loc, scale=scale, size=n_meses))
        elif distr_param[i]['mejor_dist'] == 'uniform':
            a=distr_param[i]['mejor_params']['loc']
            b= a + distr_param[i]['mejor_params']['scale']
            estimaciones[i][j]=(np.random.uniform(a, b, size=n_meses))
        elif distr_param[i]['mejor_dist'] == 'triang':
            c = distr_param[i]['mejor_params']['forma'][0]
            loc = distr_param[i]['mejor_params']['loc']
            scale = distr_param[i]['mejor_params']['scale']
            estimaciones[i][j]=(weibull_min.rvs(triang.rvs(c, loc, scale, size=n_meses)))

    return(estimaciones, n_escenarios)
```

Por medio de estima_demanda se hacen distintas estimaciones de la demanda para el horizonte de planificación. La demanda estimada dependerá de la distribución estadística de cada artículo y sus parámetros. Podemos determinar cuantas simulaciones diferentes realizar a través de n_escenarios y a través de la lista estimaciones se recogen estas simulaciones realizadas. Para esta función necesitamos algunas librerías adicionales de scipy y de la librería numpy: **import numpy as np, import random, from scipy.stats import expon, weibull_min, triang, lognorm**

```
def est_final_demandas(estimaciones, n_escenarios, articulos, meses):
    est_demandas = {}
    for i in articulos:
        sub_data = {}
        for sub_key, array in estimaciones[i].items():
            demanda_array = np.maximum(np.round(array), 0)
            sub_data[sub_key] = demanda_array
```

```
            est_demandas[i] = sub_data


    #print(est_demandas)
    return(est_demandas)
```

La función est_final_demandas sirve para depurar las estimaciones que se han realizado en la anterior función. Dado que las funciones aleatorias estadísticas pueden generar valores de demanda que no son enteros o son negativos, a partir de esta función, se redondea al entero más cercano y se pone a cero cualquier demanda estimada negativa, de manera que las estimaciones sean números enteros y mayores o iguales que cero.

```
def objetivo(simulaciones_finales, articulos, stock_minimo, n_escenarios, meses, stocks, consumo_hist, precio_art):
    factor=0.15
    rupturas=0
    rupturas={}
    inv={}
    Valor_coste_rupt={}
    Valor_coste_alm={}
    for art in articulos:
        #print("Articulo", art)
        historico_consumo=consumo_hist[art]
        demanda_mensual_promedio = historico_consumo / len(meses)
        meses_stock=2 #Es el parámetro que define la política de aprovisionamiento
        lote = round(demanda_mensual_promedio*meses_stock)
        #print("Lote de pedido", lote)
        Valor_coste_rupt[art]={}
        Valor_coste_alm[art]={}
        for i in stock_minimo:
            #print("Nuevo stock minimo ", i)
            Valor_coste_rupt[art][i]=[]
```

```
Valor_coste_alm[art][i]=[]
for j in range(n_escenarios):
    inv[j]=[]
    rupturas[j]=[]
    stock_actual=stocks[art]
    #print("Stock inicial: ", stock_actual)
    for k in range(len(meses)):
        inv[j].append(stock_actual)
        #print("inventario:", inv)
        stock_actual=stock_actual-simulaciones_finales[art][j][k]
        #print("Stock actual: ",stock_actual)
        if stock_actual<0:
            rupturas_mes=stock_actual*(-1)
            pedido_min=rupturas_mes+i
            stock_actual=stock_actual+pedido_min+lote
        elif stock_actual<i:
            pedido_min=i-stock_actual
            stock_actual=stock_actual+pedido_min+lote
        elif stock_actual>0:
            rupturas_mes=0
        rupturas[j].append(rupturas_mes)
#Calculo los valores medios de los escenarios simulados
inv_med, rupturas_promedio=valores_medios(meses, n_escenarios, rupturas, inv)
#Calculo el coste de ruptura mensual en función de las rupturas promedio
coste_ruptura_mensual=calcula_coste_ruptura(rupturas_promedio)
#Calculo el coste de ruptura en el horizonte temporal
Valor_coste_rupt[art][i].append(len(meses)*coste_ruptura_mensual)
```

```
            #Calculo el coste de almacenamiento en el horizonte temporal
            Valor_coste_alm[art][i].append(inv_med*len(meses)*precio_art[art]*factor)
    dibujar_graficas(articulos, Valor_coste_alm, Valor_coste_rupt)


    return (Valor_coste_alm, Valor_coste_rupt)
```

Mediante objetivo, se lleva a cabo el punto 4 de la metodología equivalente a la simulación para cada artículo del horizonte de planificación. Además, dentro de esta función se llama a otras funciones que nos permitirán calcular la función de costes de ruptura y representar todas las funciones de costes:

```
def valores_medios(meses, n_escenarios, rupturas, inv):
    #print("Rupturas",rupturas)
    medias_sub_inv=[]
    suma_sub_rup=[]
    #Calculo el inventario medio de cada escenario
    for sublistas in inv.values():
        media_sublista = sum(sublistas) / len(sublistas)
        medias_sub_inv.append(media_sublista)
    #Calculo la suma de rupturas de cada escenario
    for sublistas in rupturas.values():
        suma_sublista = sum(sublistas)
        suma_sub_rup.append(suma_sublista)


    # Calcular la media total de las listas
    inv_med = sum(medias_sub_inv) / len(medias_sub_inv)
    rupturas_promedio = sum(suma_sub_rup) / len(suma_sub_rup)


    return(inv_med, rupturas_promedio)
```

La función valores_medios permita obtener el valor medio de inventario y número de roturas a lo largo del horizonte temporal. Como se ha dicho, esta función es llamada desde la función objetivo y pasándole las roturas y los inventarios en cada periodo del horizonte devuelve estos valores medios con los que se obtienen las funciones de coste.

```
def calcula_coste_ruptura(rupturas_medias):
    penalizacion_max=3315
    max_caducadas=27
    if rupturas_medias<max_caducadas:
        coste_ruptura=(rupturas_medias*penalizacion_max)/max_caducadas
    else:
        coste_ruptura=penalizacion_max
    return(coste_ruptura)
```

El coste de rotura es algo más peculiar en su cálculo, para ello se implementa la función calcula_coste_ruptura. El coste de almacenamiento se calcula directamente desde la función objetivo.

```
def dibujar_graficas(articulos, almacenamiento, ruptura):
    for articulos, datos_alm in almacenamiento.items():
        datos_rupt = ruptura[articulos]
        escenarios = list(datos_alm.keys())
        coste_alm = [datos_alm[clave][0] for clave in escenarios]
        coste_rupt = [datos_rupt[clave][0] for clave in escenarios]
        coste_total = [a + b for a, b in zip(coste_alm, coste_rupt)]


        plt.plot(escenarios, coste_alm, label='Coste Almacenamiento', color='blue')
        plt.plot(escenarios, coste_rupt, label='Coste Ruptura', color='green')
        plt.plot(escenarios, coste_total, label='Coste Total', color='red')


        plt.xlabel('Escenarios de stock mínimo')
        plt.ylabel('Funciones de costes')
        plt.title(f'Gráficas para el artículo {articulos}')
```

```
        plt.legend()
        plt.show()
```

Con la función dibujar_graficas se representan las funciones de costes para los artículos.

```
def devuelve_valor_optimo(articulos, coste_almacenamiento, coste_roturas):
    valor_ss_optimo = {}
    costes = {}

    for articulo in articulos:
        coste_minimo = float('inf')  # Inicializar el costo mínimo con un valor infinito
        ss_optimo = None
        mejor_coste_almacenamiento = None
        mejor_coste_ruptura = None

        for stock_seguridad in coste_almacenamiento[articulo]:
            costo_almacenamiento = coste_almacenamiento[articulo][stock_seguridad][0]
            costo_rotura = coste_roturas[articulo][stock_seguridad][0]
            coste_total = costo_almacenamiento + costo_rotura

            if coste_total < coste_minimo:
                coste_minimo = coste_total
                ss_optimo = stock_seguridad
                mejor_coste_almacenamiento = costo_almacenamiento
                mejor_coste_ruptura = costo_rotura

        valor_ss_optimo[articulo] = ss_optimo
        costes[articulo] = {
            'mejor_coste_almacenamiento': mejor_coste_almacenamiento,
```

```
            'mejor_coste_ruptura': mejor_coste_ruptura,
            'coste_total': coste_minimo
        }

    return (valor_ss_optimo, costes)
```

Esta función llamada devuelve_valor_optimo es una forma de recopilar los datos de stock mínimo óptimo de cada artículo y el coste que se ha obtenido en las simulaciones.

```
#MAIN DEL CÓDIGO

filename= "Datos.xlsx"
datos, demanda, articulos, meses = read_file_excel_datos_demanda(filename)
datos_articulos,stocks,consumo_hco,precio_art = read_file_excel_datos_articulos(filename)
art_distr, articulos_descartados, articulos_finales=distribucion_estadistica(demanda, articulos)
estimaciones, n_escenarios=estima_demanda(art_distr, articulos_finales, meses, demanda)
estimaciones_finales=est_final_demandas(estimaciones, n_escenarios, articulos_finales, meses)
stock_minimo= list(range(101))
coste_almacenamiento,coste_roturas=objetivo(estimaciones_finales, articulos_finales, stock_minimo, n_escenarios, meses, stocks, consumo_hco, precio_art)
stock_minimo_optimo, costes_optimo = devuelve_valor_optimo(articulos_finales, coste_almacenamiento, coste_roturas)
print("Stock mínimo óptimo para cada artículo", stock_minimo_optimo)
print("Funciones de costes en el óptimo para cada artículo", costes_optimo)
```

El main con el orden de llamada a las funciones codificadas se muestra en el código de arriba. El orden es el que se indica en la metodología: lectura datos, ajusta distribución, estima demanda, simula horizonte de planificación, calcula costes y representa gráficamente.

Finalmente, se muestran dos funciones a partir de las cuales se guardan los resultados obtenidos y los datos con las demandas estimadas de la fase 1 de la metodología (se incluye también la llamada a las funciones):

```
def guardar_resultados_excel(valor_optimo, costes):
    # Creamos un libro de Excel
    libro = Workbook()
    hoja = libro.active

    # Se indican primero los encabezados que se quieren mostrar
    hoja['A1'] = 'Artículo'
    hoja['B1'] = 'Stock Mínimo Óptimo'
    hoja['C1'] = 'Coste Almacenamiento'
    hoja['D1'] = 'Coste Rotura'
    hoja['E1'] = 'Coste Total'

    # Rellenamos los datos en la hoja
    fila = 2  # Comenzamos en la fila 2 (después de los encabezados)
    for articulo, stock_minimo in valor_optimo.items():
        coste = costes[articulo]
        hoja.cell(row=fila, column=1, value=articulo)
        hoja.cell(row=fila, column=2, value=stock_minimo)
        hoja.cell(row=fila, column=3, value=coste['mejor_coste_almacenamiento'])
        hoja.cell(row=fila, column=4, value=coste['mejor_coste_ruptura'])
        hoja.cell(row=fila, column=5, value=coste['coste_total'])
        fila += 1

    # Guardar el libro en un archivo
    libro.save('RESULTADOS_PASO1.xlsx')

guardar_resultados_excel(stock_minimo_optimo, costes_optimo)
```

```
def guardar_datos_excel(datos, nombre_hoja, nombre_archivo):
    libro = load_workbook(nombre_archivo)

    # Crear una nueva hoja o obtener una existente
    if nombre_hoja in libro.sheetnames:
        hoja = libro[nombre_hoja]
    else:
        hoja = libro.create_sheet(title=nombre_hoja)

    # Escribir los encabezados
    hoja.cell(row=1, column=1, value="Artículo")
    hoja.cell(row=1, column=2, value="Escenario")



    # Escribir los datos en la hoja
    fila = 2
    for articulo, escenarios in datos.items():
        for escenario, valores in escenarios.items():
            hoja.cell(row=fila, column=1, value=articulo)
            hoja.cell(row=fila, column=2, value=escenario)
            for col, valor in enumerate(valores, start=3):
                hoja.cell(row=fila, column=col, value=valor)
            fila += 1

    # Guardar los cambios en el archivo
    libro.save(nombre_archivo)
```

```
nombre_archivo = 'RESULTADOS_PASO1.xlsx'

nombre_hoja_datos = 'DATOS_PASO1'


guardar_datos_excel(estimaciones_finales, nombre_hoja_datos, nombre_archivo)
```

El guardado de estos datos se lleva a cabo a través de la librería **from openpyxl import Workbook, load_workbook**

Para terminar este capítulo, se añaden también las funciones codificadas en la fase 2 de la metodología. Esta fase se ejecuta a través de otra rutina de Python distinta y en ella se ejecutan algunas funciones similares a las de la fase 1, pero con ciertas modificaciones:

```
def lectura_datos_excel(nombre_hoja, nombre_archivo):
    # Cargar el archivo Excel en un DataFrame
    df = pd.read_excel(nombre_archivo, sheet_name=nombre_hoja)

    # Procesar los datos y convertirlos en el formato deseado
    estimaciones_finales = {}

    for _, row in df.iterrows():
        articulo = row['Artículo']
        escenario = row['Escenario']
        valores = row.drop(['Artículo', 'Escenario']).values

        if articulo not in estimaciones_finales:
            estimaciones_finales[articulo] = {}

        estimaciones_finales[articulo][escenario] = valores

    return estimaciones_finales
```

```
def lectura_stocks_minimos(nombre_archivo, nombre_hoja):
    stocks_minimos = {}
    wb = load_workbook(nombre_archivo, read_only=True)
    hoja = wb[nombre_hoja]

    for fila in hoja.iter_rows(min_row=2, max_row=25, values_only=True):
        articulo = fila[0]
        stock_minimo = fila[1]
        stocks_minimos[articulo] = stock_minimo

    wb.close()

    return stocks_minimos
```

A través de las funciones lectura_datos_excel y lectura_stocks_minimos se leen los escenarios a simular de demanda estimada y los niveles de stock mínimo calculados en la primera fase de la metodología. Se incluye la lectura de datos a través de la carga de datos a un DataFrame para lo que hace falta la librería **import pandas as pd**

A continuación, se lleva a cabo la simulación de los distintos escenarios haciendo variaciones sobre el parámetro meses de stock. Las funciones valores_medios, calcula_coste_ruptura y devuelve_valor_optimio son idénticas a las que se definieron en la primera fase, pero las funciones objetivo y dibujar_graficas tienen sus diferencias:

```
def objetivo(meses_stock,simulaciones_finales, articulos, stocks_minimos, n_escenarios, meses, stocks, consumo_hist, precio_art):
    factor=0.15
    rupturas=0
    rupturas={}
    inv={}
    Valor_coste_rupt={}
    Valor_coste_alm={}
    for art in articulos:
```

```
#print("Articulo", art)
historico_consumo=consumo_hist[art]
demanda_mensual_promedio = historico_consumo / len(meses)
stock_minimo=stocks_minimos[art]
#meses_stock=2 #Es el parámetro que define la política de aprovisionamiento
#print("Lote de pedido", lote)
Valor_coste_rupt[art]={}
Valor_coste_alm[art]={}
for i in meses_stock:
    #print("Nuevo stcok minimo ", i)
    lote = round(demanda_mensual_promedio*i)
    Valor_coste_rupt[art][i]=[]
    Valor_coste_alm[art][i]=[]
    for j in range(n_escenarios):
        inv[j]=[]
        rupturas[j]=[]
        stock_actual=stocks[art]
        #print("Stock inicial: ", stock_actual)
        for k in range(len(meses)):
            inv[j].append(stock_actual)
            #print("inventario:", inv)
            stock_actual=stock_actual-simulaciones_finales[art][j][k]
            #print("Stock actual: ",stock_actual)
            if stock_actual<0:
                rupturas_mes=stock_actual*(-1)
                pedido_min=rupturas_mes+stock_minimo
                stock_actual=stock_actual+pedido_min+lote
```

```
            elif stock_actual<stock_minimo:
                pedido_min=i-stock_actual
                stock_actual=stock_actual+pedido_min+lote
            elif stock_actual>0:
                rupturas_mes=0
            rupturas[j].append(rupturas_mes)
        #Calculo los valores medios de los escenarios simulados
        inv_med, rupturas_promedio=valores_medios(meses, n_escenarios, rupturas, inv)
        #Calculo el coste de ruptura mensual en función de las rupturas promedio
        coste_ruptura_mensual=calcula_coste_ruptura(rupturas_promedio)
        #Calculo el coste de ruptura en el horizonte temporal
        Valor_coste_rupt[art][i].append(len(meses)*coste_ruptura_mensual)
        #Calculo el coste de almacenamiento en el horizonte temporal
        Valor_coste_alm[art][i].append(inv_med*len(meses)*precio_art[art]*factor)
    dibujar_graficas(articulos, Valor_coste_alm, Valor_coste_rupt)


    return (Valor_coste_alm, Valor_coste_rupt)
```

La función para la representación gráfica es prácticamente análoga, solamente cambia la etiqueta del eje horizontal y sus valores:

```
def dibujar_graficas(articulos, almacenamiento, ruptura):
    for articulos, datos_alm in almacenamiento.items():
        datos_rupt = ruptura[articulos]
        escenarios = list(datos_alm.keys())
        coste_alm = [datos_alm[clave][0] for clave in escenarios]
        coste_rupt = [datos_rupt[clave][0] for clave in escenarios]
        coste_total = [a + b for a, b in zip(coste_alm, coste_rupt)]


        plt.plot(escenarios, coste_alm, label='Coste Almacenamiento', color='blue')
```

```
    plt.plot(escenarios, coste_rupt, label='Coste Ruptura', color='green')
    plt.plot(escenarios, coste_total, label='Coste Total', color='red')


    plt.xlabel('Escenarios de meses de stock')
    plt.ylabel('Funciones de costes')
    plt.title(f'Gráficas para el artículo {articulos}')
    plt.legend()
    plt.show()
```

A continuación, se muestra el main del código con el orden de llamada a funciones:

```
#MAIN DEL CÓDIGO


filename= "Datos.xlsx"
datos, demanda, articulos, meses = read_file_excel_datos_demanda(filename)
datos_articulos,stocks,consumo_hco,precio_art = read_file_excel_datos_articulos(filename)
n_escenarios=10
# Nombre de la hoja y del archivo Excel
nombre_hoja = 'DATOS_PASO1'
nombre_archivo = 'RESULTADOS_PASO1.xlsx'
# Leer los datos del archivo Excel
estimaciones_finales = lectura_datos_excel(nombre_hoja, nombre_archivo)
articulos_finales= list(estimaciones_finales.keys())
# Uso de la función
name_sheet='Sheet'
stocks_minimos = lectura_stocks_minimos(nombre_archivo, name_sheet)
meses_stock=np.arange(0.1, 2.1, 0.1).tolist()
coste_almacenamiento,coste_roturas=objetivo(meses_stock, estimaciones_finales,
articulos_finales, stocks_minimos, n_escenarios, meses, stocks, consumo_hco, precio_art)
```

```
meses_stock_optimo, costes_optimo = devuelve_valor_optimo(articulos_finales, coste_almacenamiento, coste_roturas)

print("Meses de stock óptimos para cada artículo", meses_stock_optimo)

print("Funciones de costes en el óptimo para cada artículo", costes_optimo)
```

Para finalizar, se guardan los resultados de la fase 2 de la metodología en otro fichero de Excel:

```
def guardar_resultados_excel(valor_optimo, costes):
    # Creamos un libro de Excel
    libro = Workbook()
    hoja = libro.active

    # Se indican primero los encabezados que se quieren mostrar
    hoja['A1'] = 'Artículo'
    hoja['B1'] = 'Meses de stock óptimos'
    hoja['C1'] = 'Coste Almacenamiento'
    hoja['D1'] = 'Coste Rotura'
    hoja['E1'] = 'Coste Total'

    # Rellenamos los datos en la hoja
    fila = 2  # Comenzamos en la fila 2 (después de los encabezados)
    for articulo, stock_minimo in valor_optimo.items():
        coste = costes[articulo]
        hoja.cell(row=fila, column=1, value=articulo)
        hoja.cell(row=fila, column=2, value=stock_minimo)
        hoja.cell(row=fila, column=3, value=coste['mejor_coste_almacenamiento'])
        hoja.cell(row=fila, column=4, value=coste['mejor_coste_ruptura'])
        hoja.cell(row=fila, column=5, value=coste['coste_total'])
        fila += 1
```

```
    # Guardar el libro en un archivo
    libro.save('RESULTADOS_PASO2.xlsx')

guardar_resultados_excel(meses_stock_optimo, costes_optimo)
```

5 ANÁLISIS DE LOS RESULTADOS

En este apartado se detallan los resultados obtenidos en la simulación realizada. El análisis estará enfocado con preferencia a buscar el menor coste de rotura, pero siempre considerando el coste por almacenamiento. En primer lugar, se presentarán los resultados de la primera fase de la metodología de manera genérica y se mostrarán los gráficos de cada artículo para posteriormente agruparlos según el tipo de gráfico de cada artículo y analizar de manera conjunta las conclusiones de dichas representaciones tipo. Finalmente, se pasará a comentar los resultados de la segunda fase con los que se pretende hacer un análisis de sensibilidad en cada artículo para la determinación de su política de aprovisionamiento más adecuada.

5.1. Resultados primera fase de la metodología

Tabla 9: Resultados fase 1

Artículo	Descripcion	Stock Mínimo Óptimo	Coste Almacenamiento	Coste Rotura	Coste Total
B61425000184	LPB SCII CPU RETAIL-SC-II	8	14.447,51 €	7.956,00 €	22.403,51 €
B61431051020	IMPRESORA XC	6	7.795,50 €	6.077,50 €	13.873,00 €
B61421060003	LPB SCII F.A.	13	10.865,21 €	3.315,00 €	14.180,21 €
B71058420000	CABEZAL TERMICO SC	18	8.293,73 €	773,50 €	9.067,23 €
DBCES0126F	MONTAJE ETIQUETADORA	10	6.002,84 €	15.691,00 €	21.693,84 €
DBH-KF2002GR	CABEZAL IMPRESIÓN	36	6.429,17 €	23.315,50 €	29.744,67 €
E000598002339	CABEZAL ETIQ. PT562 TPH 60MM	33	7.751,18 €	13.149,50 €	20.900,68 €
B61421040701	TECLADO SC II N	13	7.646,00 €	8.729,50 €	16.375,50 €
B61340227100	WS18 C6/06 C3MI/06 ADW501	8	5.311,60 €	5.083,00 €	10.394,60 €
B61425105154	DISPLAY SOLOMON AMBER V1.54	8	4.676,54 €	331,50 €	5.008,04 €
E000598002331	ETIQUETADORA DE 60 MM 1061	8	3.876,46 €	3.204,50 €	7.080,96 €
E00019016760	PLACA PC PICO ITX (ACTUALIZADA)	3	2.872,56 €	4.199,00 €	7.071,56 €
D4501003792R	CPU L8/L5 T O ETIQ. BOX/SCA 1UART8X	4	4.400,31 €	1.326,00 €	5.726,31 €
B61421602001	IMPRESORA SCII-BCII 100	4	2.867,76 €	- €	2.867,76 €
B61425020174	LPB SC II HW 7" CPU 1.74	2	2.886,10 €	1.547,00 €	4.433,10 €
B61431630021	PARTE SUP CARCASA CPL IMPRES LINEAS 2	6	3.042,08 €	7.845,50 €	10.887,58 €
E000119252132	FAC GAMA ALTA VENTA	6	1.942,90 €	- €	1.942,90 €
B71058410000	CABEZAL IMPRESORA TERMICA 2003-CF11B	0	4.663,64 €	- €	4.663,64 €
B61431620020	ESTRUCTURA DE BASE IMPRESORA XC 720	3	1.403,14 €	331,50 €	1.734,64 €
DBCES0146C	MONTAJE ETIQUETADORA CS1100/1200	4	2.187,10 €	2.762,50 €	4.949,60 €
E000119309800	P.G.GA ETHERNET NF	2	1.244,40 €	1.768,00 €	3.012,40 €
D4501003732R	CPU L8/L5 ETHERNET + 2 UARTS 8X	2	1.475,64 €	1.989,00 €	3.464,64 €
E000598002351	KIT PRT CABEZAL TPH-791	8	2.074,25 €	- €	2.074,25 €
B61431610020	PUERTA SIN REBOBINADOR XC KPL	4	1.512,97 €	331,50 €	1.844,47 €

Según los resultados obtenidos, los artículos que más coste de rotura supondrían son el montaje etiquetadora DBCES0126F y los cabezales de impresión DBH-KF2002GR y E000598002339. Estos elevados costes de rotura se deben principalmente al bajo nivel de stock de estos artículos al comienzo de las simulaciones.

Dado que la demanda estimada en cada periodo, de media, es mayor que el stock con el que se parte la simulación, se incurren costes de rotura muy elevados.

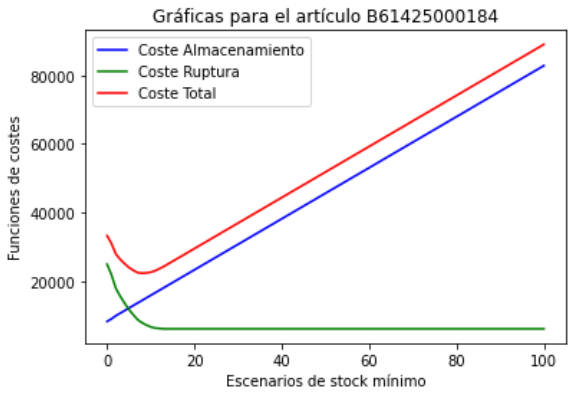
Esta misma situación ocurre con la placa base B61425000184, la impresora XC B61421051020, el teclado SC II N B61421040701, la célula de pesaje WS18 C6/06 C3MI/06 ADW501 B61240227100, la placa E00019016760, el montaje etiquetadora DBCES0146C, la placa E000119309800, o la CPU D4501003732R.

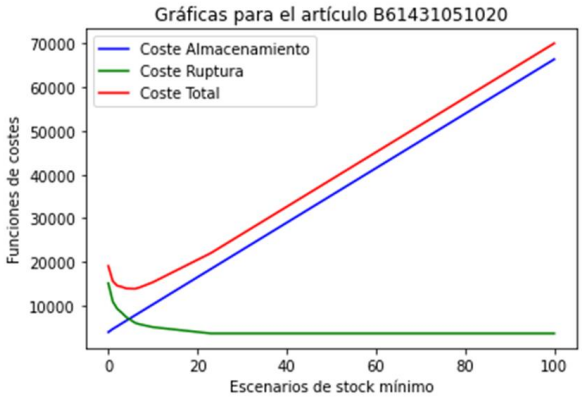
Todos los artículos mencionados anteriormente poseen un coste de rotura inadmisible si perseguimos un nivel de servicio excelente para nuestro cliente. El coste de rotura de estos artículos depende del nivel de demanda esperado de cada uno, así a mayor demanda esperada mayor coste de rotura. De esta forma, podemos agrupar el primer tipo de artículos con la categoría de referencias a comprar dado que con el bajo stock que partimos ahora mismo no es suficiente para cubrir la demanda esperada:

Tabla 10: Artículos clasificados tipo 1: recomendación de compra

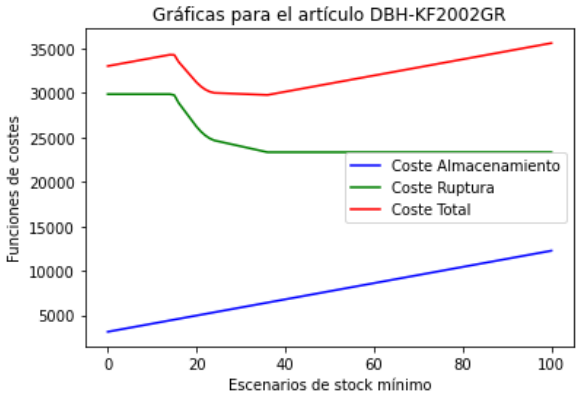
Artículo	Descripcion	Stock Mínimo Óptimo	Stock Inicial	Unidades a comprar
B61425000184	LPB SCII CPU RETAIL-SC-II	8	2	6
B61431051020	IMPRESORA XC	6	3	3
DBCES0126F	MONTAJE ETIQUETADORA	10	3	7
DBH-KF2002GR	CABEZAL IMPRESIÓN	36	4	32
E000598002339	CABEZAL ETIQ. PT562 TPH 60MM	33	5	28
B61421040701	TECLADO SC II N	13	1	12
B61340227100	WS18 C6/06 C3MI/06 ADW501	8	0	8
E00019016760	PLACA PC PICO ITX (ACTUALIZADA)	3	0	3
DBCES0146C	MONTAJE ETIQUETADORA CS1100/1200	4	0	4
E000119309800	P.G.GA ETHERNET NF	2	0	2
D4501003732R	CPU L8/L5 ETHERNET + 2 UARTS 8X	2	0	2

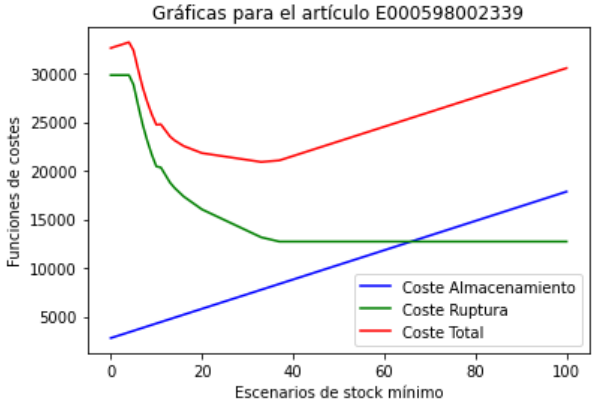
Estos artículos tienen las siguientes curvas de costes:

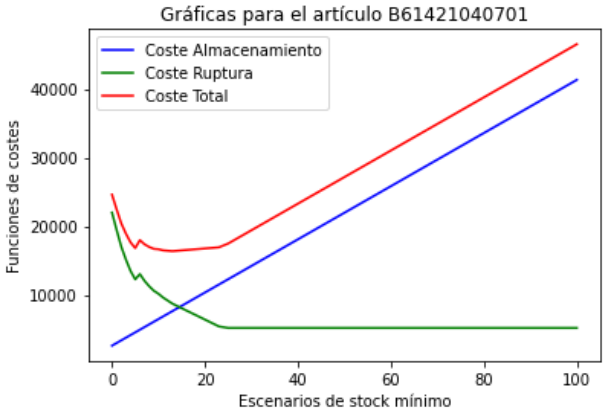


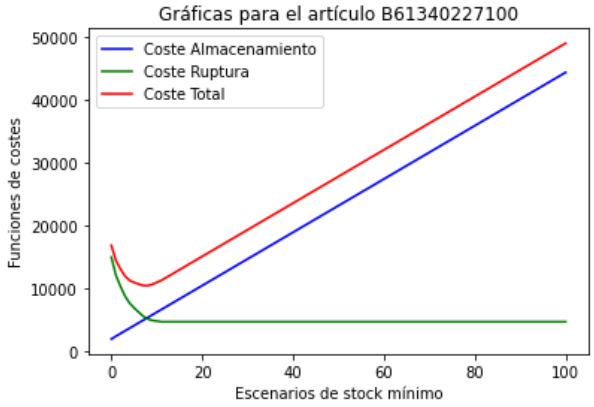



Gráficas para el artículo DBCES0126F
Coste Almacenamiento
Coste Ruptura
Coste Total
Funciones de costes
Escenarios de stock mínimo

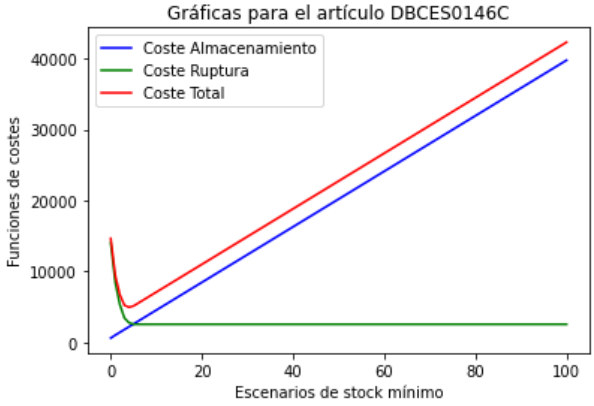

Gráficas para el artículo DBH-KF2002GR
Coste Almacenamiento
Coste Ruptura
Coste Total
Funciones de costes
Escenarios de stock mínimo


Gráficas para el artículo E000598002339
Coste Almacenamiento
Coste Ruptura
Coste Total
Funciones de costes
Escenarios de stock mínimo

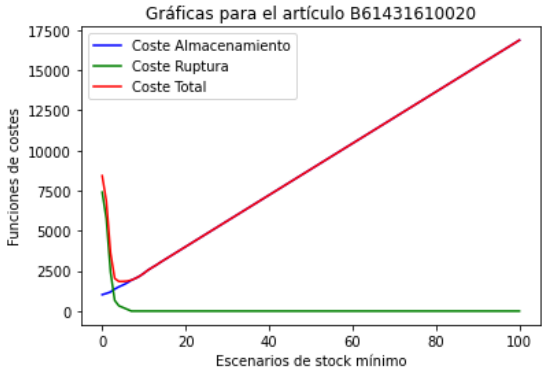

Gráficas para el artículo B61421040701
Coste Almacenamiento
Coste Ruptura
Coste Total
Funciones de costes
Escenarios de stock mínimo


Gráficas para el artículo B61340227100
Coste Almacenamiento
Coste Ruptura
Coste Total
Funciones de costes
Escenarios de stock mínimo


Gráficas para el artículo E00019016760
Coste Almacenamiento
Coste Ruptura
Coste Total
Funciones de costes
Escenarios de stock mínimo


Gráficas para el artículo DBCES0146C
Coste Almacenamiento
Coste Ruptura
Coste Total
Funciones de costes
Escenarios de stock mínimo


Gráficas para el artículo E000598002351
Coste Almacenamiento
Coste Ruptura
Coste Total
Funciones de costes
Escenarios de stock mínimo



En general, todas las gráficas tienen una forma muy similar a excepción de dos de los artículos: DBCES0126F y B61421040701. Estas dos referencias mejoran el coste de rotura para niveles de stock mínimo por encima de las 20 unidades, pero como el coste de almacenamiento es mayor, el óptimo se encuentra en los puntos ya comentados.

Los resultados obtenidos para estos artículos no son lo suficientemente concluyentes para determinar con buen criterio el stock mínimo por lo que se comentaba anteriormente del bajo nivel de stock que tienen al comienzo de la simulación. Con el fin de poder sacar alguna conclusión más convincente, se propone la realización de las simulaciones partiendo del stock mínimo óptimo que se ha obtenido tras correr la primera fase de la metodología, en lugar de los stocks reales. Los resultados de stock mínimo obtenidos y los costes en el óptimo son los siguientes:

Tabla 11: Resultados fase 1, mejoras artículos tipo 1

Artículo	Descripcion	Stock Mínimo Óptimo	Coste Almacenamiento	Coste Rotura	Coste Total	Mejora
B61425000184	LPB SCII CPU RETAIL-SC-II	8	15.116,46 €	5.525,00 €	20.641,46 €	1.762,05 €
B61431051020	IMPRESORA XC	4	6.556,02 €	7.293,00 €	13.849,02 €	23,98 €
DBCES0126F	MONTAJE ETIQUETADORA	11	6.665,10 €	2.099,50 €	8.764,60 €	12.929,24 €
DBH-KF2002GR	CABEZAL IMPRESIÓN	43	7.226,70 €	2.210,00 €	9.436,70 €	20.307,97 €
E000598002339	CABEZAL ETIQ. PT562 TPH 60MM	36	8.973,66 €	1.215,50 €	10.189,16 €	10.711,52 €
B61421040701	TECLADO SC II N	9	6.659,42 €	3.315,00 €	9.974,42 €	6.401,08 €
B61340227100	WS18 C6/06 C3MI/06 ADW501	5	4.256,70 €	1.326,00 €	5.582,70 €	4.811,90 €
E00019016760	PLACA PC PICO ITX (ACTUALIZADA)	5	4.098,73 €	2.873,00 €	6.971,73 €	99,84 €
DBCES0146C	MONTAJE ETIQUETADORA CS1100/1200	4	2.304,53 €	442,00 €	2.746,53 €	2.203,07 €
E000119309800	P.G.GA ETHERNET NF	2	1.321,92 €	552,50 €	1.874,42 €	1.137,98 €
D4501003732R	CPU L8/L5 ETHERNET + 2 UARTS 8X	2	1.502,27 €	994,50 €	2.496,77 €	967,86 €

Estos nuevos resultados mejoran la función global de coste en unos 60.000 €, casi todo gracias a un menor coste de rotura. Además, los valores de stock mínimo óptimo son muy similares a los que se obtuvieron en la anterior prueba lo que demuestra que el coste de rotura anterior tan elevado se debía al nivel de stock mínimo al comienzo de la simulación.

Antes de analizar los demás artículos del estudio hay que resaltar que el listado de referencias que se simulaban en la rutina de Python era de 25, sin embargo, sólo se han obtenido resultados de 24 de los artículos. Esto se debe a que uno de los artículos no ha superado la prueba de Kolmogórov-Smirnov y, por tanto, ha quedado fuera del estudio. Se trata de la referencia B61431079002, una pantalla táctil que apenas se ha consumido en 2 ocasiones en los últimos 9 meses. La función de ajuste estadístico considera que no hay datos suficientes de demanda como para identificar una distribución apta.

Se muestra, a continuación, la tabla con los resultados para el resto de los artículos señalando en color rojo aquellos que merecen un comentario adicional:

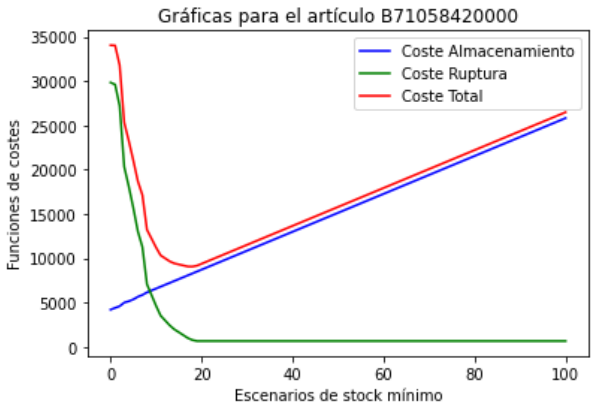
Tabla 12: Artículos clasificados tipo 2

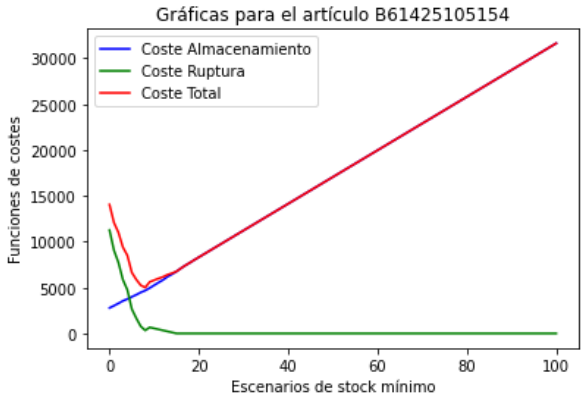
Artículo	Descripcion	Stock Mínimo Óptimo	Coste Almacenamiento	Coste Rotura	Coste Total
B61421060003	LPB SCII F.A.	13	10.865,21 €	3.315,00 €	14.180,21 €
B71058420000	CABEZAL TERMICO SC	18	8.293,73 €	773,50 €	9.067,23 €
B61425105154	DISPLAY SOLOMON AMBER V1.54	8	4.676,54 €	331,50 €	5.008,04 €
E000598002331	ETIQUETADORA DE 60 MM 1061	8	3.876,46 €	3.204,50 €	7.080,96 €
D4501003792R	CPU L8/L5 T O ETIQ. BOX/SCA 1UART8X	4	4.400,31 €	1.326,00 €	5.726,31 €
B61421602001	IMPRESORA SCII-BCII 100	4	2.867,76 €	- €	2.867,76 €
B61425020174	LPB SC II HW 7" CPU 1.74	2	2.886,10 €	1.547,00 €	4.433,10 €
B61431630021	PARTE SUP CARCASA CPL IMPRES LINEAS 2	6	3.042,08 €	7.845,50 €	10.887,58 €
E000119252132	FAC GAMA ALTA VENTA	6	1.942,90 €	- €	1.942,90 €
B71058410000	CABEZAL IMPRESORA TERMICA 2003-CF11B	0	4.663,64 €	- €	4.663,64 €
B61431620020	ESTRUCTURA DE BASE IMPRESORA XC 720	3	1.403,14 €	331,50 €	1.734,64 €
E000598002351	KIT PRT CABEZAL TPH-791	8	2.074,25 €	- €	2.074,25 €
B61431610020	PUERTA SIN REBOBINADOR XC KPL	4	1.512,97 €	331,50 €	1.844,47 €

La placa base B61421060003, la etiquetadora E000598002331 y la carcasa superior B61431630031 son las que tienen mayor coste total, incluyendo un alto coste de rotura. Para estos artículos, habría que revisar la política de aprovisionamiento, es decir, analizar cuantas unidades hay que hacer cada vez que se hace un pedido para disminuir el coste de rotura. Este análisis se llevará a cabo a continuación, cuando se muestren los resultados de la segunda fase de la metodología. Otra opción es observar el gráfico por si fuera posible minimizar el coste de rotura, aunque esto suponga un aumento del coste total; lo veremos a continuación.

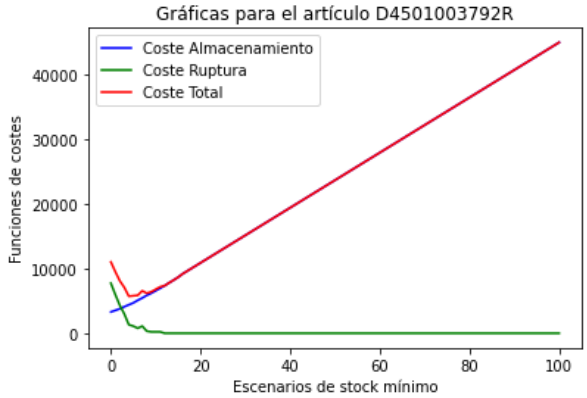
También se ha resaltado el cabezal de impresión B71058410000, para el cual se recomienda un stock mínimo de cero con el que se conseguiría un coste de rotura de 0 €. Esto se debe a que el stock actual de este artículo es tan alto que la demanda total en el horizonte de planificación nunca superará este valor de inventario haciendo que el coste sea mínimo cuando el coste de almacenamiento es mínimo y esto se produce cuando el inventario está en los niveles más bajos, lo cual se consigue con un stock mínimo de cero.

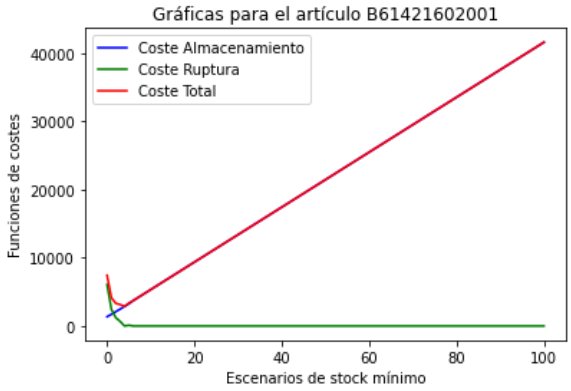
Las gráficas de costes de estos artículos se muestran a continuación:

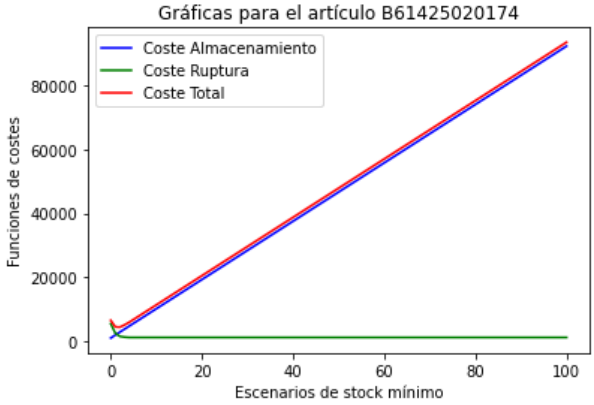


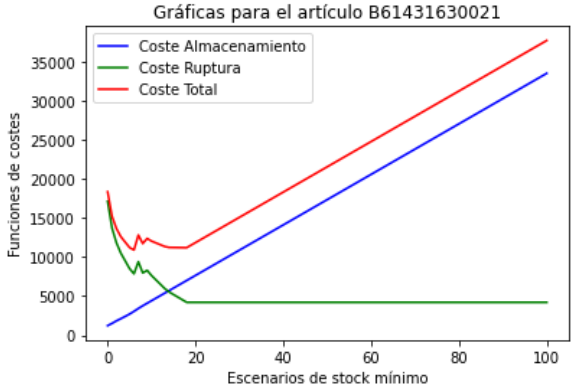

Gráficas para el artículo B61425105154
Coste Almacenamiento
Coste Ruptura
Coste Total
Funciones de costes
Escenarios de stock mínimo


Gráficas para el artículo E000598002331
Coste Almacenamiento
Coste Ruptura
Coste Total
Funciones de costes
Escenarios de stock mínimo

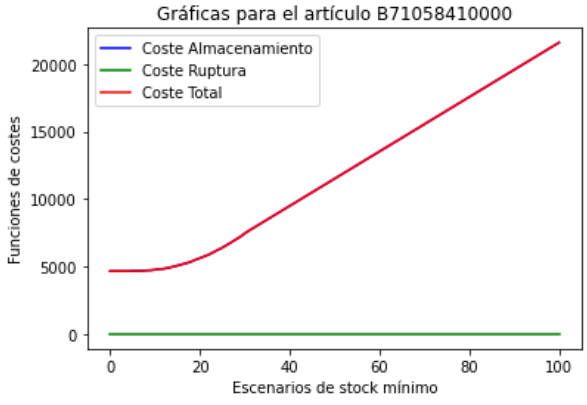

Gráficas para el artículo D4501003792R
Coste Almacenamiento
Coste Ruptura
Coste Total
Funciones de costes
Escenarios de stock mínimo

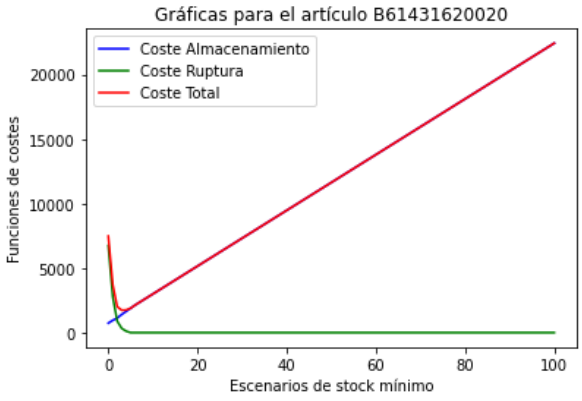

Gráficas para el artículo B61421602001
Coste Almacenamiento
Coste Ruptura
Coste Total
Funciones de costes
Escenarios de stock mínimo

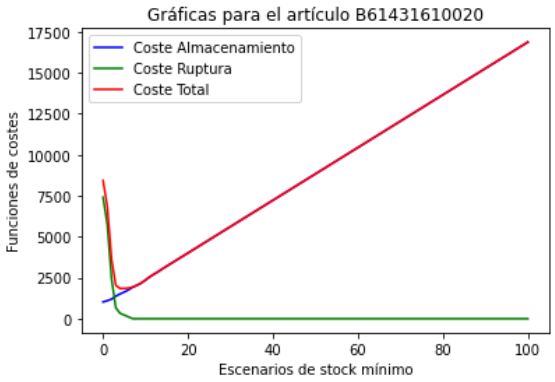

Gráficas para el artículo B61425020174
Coste Almacenamiento
Coste Ruptura
Coste Total
Funciones de costes
Escenarios de stock mínimo


Gráficas para el artículo B61431630021
Coste Almacenamiento
Coste Ruptura
Coste Total
Funciones de costes
Escenarios de stock mínimo


Gráficas para el artículo E000119252132
Coste Almacenamiento
Coste Ruptura
Coste Total
Funciones de costes
Escenarios de stock mínimo


Gráficas para el artículo B71058410000
Coste Almacenamiento
Coste Ruptura
Coste Total
Funciones de costes
Escenarios de stock mínimo





En vista a los gráficos, se puede llegar a la conclusión de que en la mayoría de los artículos podemos disminuir el coste de rotura, pero ello supondría penalizar la función objetivo global con un mayor coste de almacenamiento. Esto sucede en todos los artículos que se mencionaron antes de mostrar la representación gráfica.

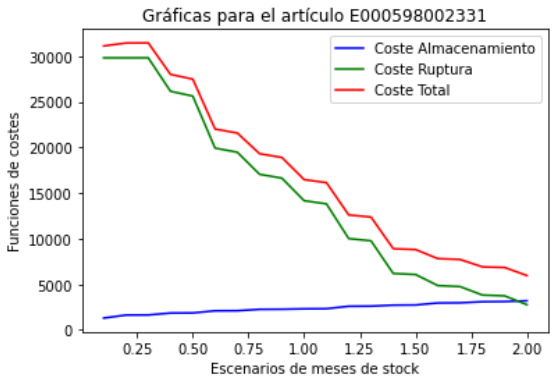
Sería posible modificar la lógica de la rutina y penalizar más el coste de rotura para evolucionar a niveles de stock mínimo más elevados que eviten las roturas de stock.

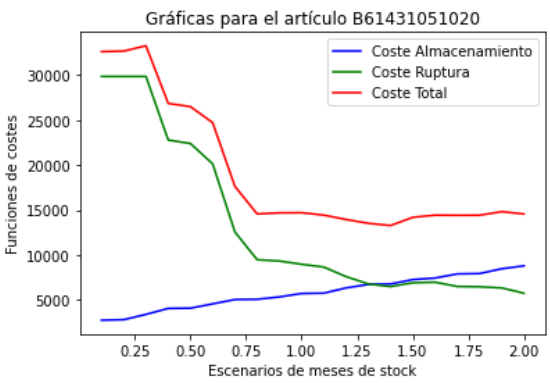
5.2. Resultados segunda fase de la metodología

Tabla 13: Resultados fase 2

Artículo	Descripcion	Meses de stock óptimos	Coste Almacenamiento	Coste Rotura	Coste Total
B61425000184	LPB SCII CPU RETAIL-SC-II	1,9	12.867,11 €	8.574,80 €	21.441,91 €
B61431051020	IMPRESORA XC	1,4	6.783,64 €	6.497,40 €	13.281,04 €
B61421060003	LPB SCII F.A.	1,9	8.049,00 €	7.502,95 €	15.551,95 €
B71058420000	CABEZAL TERMICO SC	1,8	6.221,37 €	1.701,70 €	7.923,07 €
DBCES0126F	MONTAJE ETIQUETADORA	2	5.089,14 €	20.442,50 €	25.531,64 €
DBH-KF2002GR	CABEZAL IMPRESIÓN	1,7	4.589,05 €	23.315,50 €	27.904,55 €
E000598002339	CABEZAL ETIQ. PT562 TPH 60MM	2	5.814,80 €	21.768,50 €	27.583,30 €
B61421040701	TECLADO SC II N	2	5.914,65 €	13.260,00 €	19.174,65 €
B61340227100	WS18 C6/06 C3MI/06 ADW501	2	4.219,60 €	5.746,00 €	9.965,60 €
B61425105154	DISPLAY SOLOMON AMBER V1.54	2	3.991,28 €	884,00 €	4.875,28 €
E000598002331	ETIQUETADORA DE 60 MM 1061	2	3.195,12 €	2.762,50 €	5.957,62 €
E00019016760	PLACA PC PICO ITX (ACTUALIZADA)	1,9	2.794,27 €	3.933,80 €	6.728,07 €
D4501003792R	CPU L8/L5 T O ETIQ. BOX/SCA 1UART8X	1,9	4.028,47 €	2.419,95 €	6.448,42 €
B61421602001	IMPRESORA SCII-BCII 100	1,9	2.575,94 €	55,25 €	2.631,19 €
B61425020174	LPB SC II HW 7" CPU 1.74	1,9	2.550,72 €	1.768,00 €	4.318,72 €
B61431630021	PARTE SUP CARCASA CPL IMPRES LINEAS 2	1,9	2.570,22 €	8.928,40 €	11.498,62 €
E000119252132	FAC GAMA ALTA VENTA	1,8	1.382,43 €	596,70 €	1.979,13 €
B71058410000	CABEZAL IMPRESORA TERMICA 2003-CF11B	0,1	4.663,64 €	- €	4.663,64 €
B61431620020	ESTRUCTURA DE BASE IMPRESORA XC 720	2	1.316,79 €	110,50 €	1.427,29 €
DBCES0146C	MONTAJE ETIQUETADORA CS1100/1200	2	1.732,07 €	4.574,70 €	6.306,77 €
E000119309800	P.G.GA ETHERNET NF	2	1.244,40 €	1.768,00 €	3.012,40 €
D4501003732R	CPU L8/L5 ETHERNET + 2 UARTS 8X	2	1.475,64 €	1.989,00 €	3.464,64 €
E000598002351	KIT PRT CABEZAL TPH-791	1,9	1.452,81 €	375,70 €	1.828,51 €
B61431610020	PUERTA SIN REBOBINADOR XC KPL	2	1.398,44 €	110,50 €	1.508,94 €

En vista a los resultados de la segunda fase, se podría concluir que la política de aprovisionar dos meses de stock cada vez que realizamos un pedido es prácticamente el óptimo para todos los artículos. Todos sitúan en torno a 2 el valor de meses de stock para el lote de pedido a excepción del cabezal B71058410000, que como ya se comentó anteriormente su inventario inicial era suficientemente grande como para hacer frente a la demanda estimada de todo el horizonte temporal, por lo que se entiende que los meses de stock sean lo más bajo posibles. Para comprender el comportamiento gráfico de las funciones de coste a partir de este análisis de sensibilidad, se muestran a continuación los gráficos de la etiquetadora de 60 mm E000598002331 y de la impresora XC B61431051020 que toman los datos 2 y 1,4 meses de stock, respectivamente:





6 CONCLUSIONES. PLAN DE ACCIÓN

En esta última sección se comentan las conclusiones generales que se pueden sacar de los resultados obtenidos y, además, se enumeran las acciones a tomar tras a nivel organizativo en la empresa para mejorar la calidad del sistema de gestión de almacenes que tenemos actualmente.

Se podría decir que este estudio aporta un valor importantísimo al proyecto ya que constituye una herramienta muy útil para evaluar posibles modificaciones en los niveles de inventario de los distintos artículos del almacén, así como de la política de aprovisionamiento de cada uno. Además, permite identificar necesidades de compra como se ha visto en el análisis de los resultados.

Los objetivos del proyecto se cumplen con creces dando una solución al problema de la gestión de inventarios de la empresa. Con el tiempo, se podrá comprobar si verdaderamente se consiguen optimizar los recursos de la empresa con el seguimiento económico del proyecto y se alcanza un nivel de servicio excelente gracias a la desaparición de las roturas de stock.

Adicionalmente, hay que mencionar que este estudio tendrá continuidad en el tiempo y posee un gran potencial para seguir promoviendo la mejora continua dentro de Tier1 a través de las siguientes acciones:

1. Monitorización en el tiempo del consumo de los artículos para corroborar que las estimaciones de demanda se asemejan a la realidad.
2. Sugerir la realización de las compras mencionadas en el análisis de resultados.
3. Integración en el ERP de la empresa de la lógica de niveles de inventario para actualizar los valores de stock mínimo y el punto de pedido de referencias.
4. Seguimiento económico del proyecto para verificar que las medidas implementadas están resultando efectivas.
5. Actualizar el código a las necesidades que se presenten en el día a día del proyecto. Podría interesar en un futuro realizar simulaciones de demanda semanales e incluir algunas restricciones como el tiempo de suministro según el fabricante de la referencia o variar los costes de rotura según nuevos acuerdos con el cliente.

REFERENCIAS

REFERENCIAS LITERARIAS:

[1] Abdul-Jalbar Betancor, B. (2004). Sistemas de distribución: avances en la gestión de inventarios (Tesis Doctoral). Universidad de la Laguna, Tenerife.

[2] Franco López, C. L., & Rodríguez Ramirez, A. L. (2021). PROPUESTA PARA OPTIMIZAR LA GESTIÓN DE INVENTARIOS Y SU INFLUENCIA EN LOS ESTADOS FINANCIEROS DE UNA EMPRESA COMERCIAL (Trabajo de investigación). Pontificia Universidad Católica de Perú.

[3] Companys Oascual, R. (1990). Previsión tecnológica y de la demanda. Marcombo S, A.

[4] Díaz, G. E. (2019). Previsión de la demanda de una empresa en el sector de comercio de electrodomésticos (Proyecto de fin de grado). Universidad de Sevilla.

[5] Vidal Holguín, C. J. (2017). Fundamentos de control y gestión de inventarios (Libro académico). Universidad del Valle, Colombia.

[6] Espejo González, M. (2022). Gestión de inventarios: métodos cuantitativos. Marge Books.

REFERENCIAS ELECTRÓNICAS:

[1] Tier1. (Julio 2023). Recuperado de [Enlace]: https://www.tier1.es/es/web/guest/home

[2] SciPy. (Julio 2023). scipy.stats.goodness_of_fit. Recuperado de [Enlace]: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.goodness_of_fit.html

[3] SciPy. (Julio 2023). scipy.stats.kstest. Recuperado de [Enlace]: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.kstest.html

[4] SciPy. (Julio 2023). scipy.stats.rv_continuous.rvs. Recuperado de [Enlace]: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.rv_continuous.rvs.html

[5] SciPy. (Julio 2023). scipy.stats.fit. Recuperado de [Enlace]: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.fit.html

[6] NumPy. (Julio 2023). numpy.random.normal. Recuperado de [Enlace]: https://numpy.org/doc/stable/reference/random/generated/numpy.random.normal.html