Universidad de Valladolid

FACULTAD DE CIENCIAS

TRABAJO FIN DE GRADO

Grado en Estadística

Aplicación Python para la creación de un BI desde un sistema gestor ERP

Autor: Rubén Reyes Guerrero

Tutor: Agustín Mayo Íscar

Tutor Externo: Iván Cáceres Pascual

2025

Resumen

En la actualidad, las pequeñas y medianas empresas (PYMES) enfrentan importantes desafíos a la hora de aprovechar el potencial de sus datos para la toma de decisiones estratégicas. Este Trabajo de Fin de Grado (TFG) se centra en el desarrollo de una aplicación de Business Intelligence (BI) basada en tecnologías de código abierto como Python, D3.js, HTML y CSS, diseñada específicamente para entornos empresariales con recursos limitados. La solución propuesta se integra directamente con un software ERP (Sage 200), lo que permite la extracción automática de datos contables en tiempo real y su posterior análisis y visualización mediante cuadros de mando interactivos.

El sistema desarrollado permite aplicar filtros personalizados, analizar el rendimiento comercial por agentes, productos y marcas, y realizar predicciones de ventas. Para ello, se han implementado procesos de tratamiento de datos, técnicas de anonimización y una interfaz sencilla que facilita el acceso a información clave incluso para usuarios sin formación técnica avanzada.

Este TFG está basado en un caso de uso real y utiliza datos reales de una empresa, lo que garantiza la aplicabilidad de los resultados en contextos reales. La herramienta no solo representa una alternativa económica frente a soluciones comerciales ya existentes, sino que también promueve la democratización del análisis de datos en empresas que tradicionalmente han estado al margen de la transformación digital. La propuesta aporta así un enfoque innovador y escalable para mejorar la competitividad de las PYMES a través de la analítica de datos.

Abstract

Currently, small and medium-sized enterprises (SMEs) face significant challenges when it comes to leveraging the potential of their data for strategic decision-making. This Final Degree Project (TFG) focuses on the development of a Business Intelligence (BI) application based on open-source technologies such as Python, D3.js, HTML and CSS, designed specifically for business environments with limited resources. The proposed solution integrates directly with ERP software (Sage 200), enabling the automatic extraction of accounting data in real time and its subsequent analysis and visualisation through interactive dashboards.

The system developed allows customised filters to be applied, commercial performance to be analysed by agents, products and brands, and sales forecasts to be made.

To this end, data processing procedures, anonymisation techniques and a simple interface have been implemented to facilitate access to key information, even for users without advanced technical training. This TFG is based on a real use case and uses real data from a company, which guarantees the applicability of the results in real contexts.

The tool not only represents a cost-effective alternative to existing commercial solutions, but also promotes the democratisation of data analysis in companies that have traditionally been on the sidelines of digital transformation. The proposal thus provides an innovative and scalable approach to improving the competitiveness of SMEs through data analytics.

Agradecimientos

Quiero agradecer de corazón a mi familia su apoyo incondicional y a lo largo de toda mi carrera y sobre todo con este trabajo. También a mi pareja por ser la persona que nunca me ha permitido rendirme y que siempre ha sabido encontrar las palabras justas para una dosis de motivación.

Además, quiero dar las gracias a Agustín por sus enseñanzas, su disponibilidad y su buen criterio para guiarme a lo largo de este trabajo.

Por último, quiero agradecer a Iván Cáceres por su paciencia, su apoyo tanto laboral como externo y su disposición a enseñar sin esperar nada a cambio.

Índice general

Índice de figuras

1. Introducción

En la era de la economía digital, la capacidad de transformar datos en decisiones estratégicas constituye un papel fundamental que puede marcar un diferencial competitivo para las organizaciones. El Business Intelligence (BI) surge como herramienta clave en este proceso, evolucionando desde sistemas de reporting estático en un inicio, hasta ecosistemas analíticos integrados con inteligencia artificial (Davenport, 2014). Sin embargo, este progreso no ha alcanzado a todas las empresas de la misma manera. Mientras las grandes corporaciones cuentan con herramientas avanzadas de análisis y predicción de datos , aproximadamente el 67% de las pequeñas y medianas empresas (PYMES) aún utilizan herramientas sin tanto desarrollo como hojas de cálculo para la toma de decisiones (McKinsey, 2021). Esta diferencia entre grandes y pequeñas compañías plantea un problema crítico en un mercado donde más de la mitad de la información que se tiene acerca de las organizaciones no se emplea correctamente (IDC, 2023), particularmente en compañías con recursos limitados.

La transformación digital ha redefinido las necesidades analíticas empresariales. Estudios recientes identifican tres brechas principales en las PYMES (Chen et al., 2022):

- Técnica: Muchas PYMES cuentan con soluciones genéricas que no se adaptan a procesos específicos
- Económica: Costos elevados de licencias para herramientas premium
- Operacional: Falta de personal cualificado para implementar o interpretar sistemas complejos de análisis de información

Además, el mundo tecnológico ha desarrollado capacidades sin precedentes. Plataformas como Python ofrecen librerías especializadas (Pandas, Scikit-learn) que pueden replicar funcionalidades concretas de software comercial (Van Rossum & Drake, 2023). Esta dicotomía entre necesidades por satisfacer y herramientas disponibles crea un escenario ideal para soluciones como aplicaciones hechas en python.

Actualmente el BI para PYMES se centra en dos enfoques, por un lado, existen las soluciones SaaS comerciales (Tableau, Power BI), que son eficaces para análisis básicos, pero con limitaciones en personalización y costos recurrentes (Porter & Heppelmann, 2015), y también plataformas low-code, que democratizan el acceso pero a cambio se pierde capacidad de integración con sistemas (Gartner, 2023)

Investigaciones recientes (Hoffmann, Nagle & Zhou, 2024) demuestran que el uso de personalizaciones basadas en Python podría ayudar a compañías con pocos recursos en una reducción de costos operativos frente a alternativas comerciales, sin tener que renunciar a la precisión analítica. No obstante, persiste un vacío en estos modelos que sean capaces de equilibrar flexibilidad técnica con usabilidad para perfiles no especializados.

Esta investigación propone desarrollar un framework de BI modular construido con herramientas como Python y d3, dirigido específicamente a PYMES, para ello la aplicación ha de enfocarse en lo siguiente:

1. Extracción unificada de datos estructurados y no estructurados
2. Automatización de flujos ETL mediante pipelines configurables
3. Modelado predictivo accesible mediante interfaces intuitivas
4. Visualización interactiva adaptable a distintos perfiles usuarios

La hipótesis central sostiene que una solución desarrollada con tecnologías abiertas puede superar el trilema PYMES (costo-flexibilidad-usabilidad) mejor que las alternativas comerciales existentes.

El estudio combina el análisis comparativo de varias herramientas existentes de BI en dimensiones técnicas y económicas, así como el desarrollo de un prototipo de software alternativo.

1.1. Objetivos

El presente Trabajo de Fin de Grado (TFG) tiene como propósito principal diseñar y desarrollar una aplicación software económica, fácil de implementar y orientada a cubrir necesidades operativas de pequeñas y medianas empresas (pymes), ofreciendo resultados claros y fácilmente interpretables por perfiles no técnicos.

Estos objetivos responden a la motivación inicial de ofrecer a las pymes una herramienta:

- Económica: límite de coste operativo y retorno de inversión medible.
- Útil: validación funcional en entornos reales y mejora tangible de procesos.
- Interpretables: foco en UX y dashboards comprensibles para directivos sin perfil técnico.

Objetivos de aprendizaje personal:

Profundizar en metodologías ágiles (Scrum/Kanban) aplicadas a proyectos de software de alcance limitado.

Consolidar competencias en visualización de datos (librerías D3.js, Chart.js o equivalentes) orientadas a usuarios de negocio.

Desarrollar habilidades de investigación aplicada, desde la definición del problema hasta la validación de la solución con usuarios reales.

1.2. Estructura de la memoria

La estructura de ese documento, que también se puede apreciar en el Índice General, se puede esquematizar según los siguientes capítulos:

Capítulo 1. Introducción: En este presente capítulo se introduce el proyecto de fin de grado, su contexto, motivación y objetivos, y la estructura de la actual memoria donde se documenta el proyecto.

Capítulo 2 Contexto: Durante este capítulo se trata el contexto y los conceptos claves teóricos relacionados con el tema del proyecto: los sistemas ERP, los modelos de BI ya existentes y las herramientas de programación utilizadas

Capítulo 3. Datos: En este tercer capítulo se extraen los datos y se encriptan para no comprometer la sensibilidad de los mismos y se define el input con el que se va a trabajar.

Capítulo 4. Características, ventajas y desventajas de la aplicación: En este capítulo se expone la aplicación desarrollada como solución al problema inicial se explican sus características y las principales ventajas y desventajas con respecto a otros modelos

Capítulo 5. Posibles implementaciones: Durante este capítulo se describe el trabajo a futuro y las posibilidades de mejorar la aplicación con nuevos desarrollos

Capítulo 6. Conclusiones: En el trascurso de este capítulo se presentan las conclusiones obtenidas al finalizar el proyecto.

Bibliografía: Se referencian de las fuentes de información utilizadas durante el proyecto. Anexos: Se proporciona material complementario y detallado que respalda y enriquece la comprensión del trabajo realizado. Este material consiste principalmente en el código desarrollado durante el proyecto.

2. Contexto

Las aplicaciones de Business Intelligence (BI) están diseñadas para recopilar, transformar, analizar y visualizar datos con el fin de facilitar la toma de decisiones estratégicas en las organizaciones.

Estas aplicaciones necesitan datos de diversas fuentes, que pueden incluir:

- Bases de datos relacionales (SQL Server, MySQL, PostgreSQL, Oracle, etc.).
- Sistemas transaccionales (ERP, CRM, SCM). Aunque en algunas ocasiones (como es el caso de este trabajo) los datos de estos sistemas se almacenan en una base de datos relacional.
- Archivos planos (CSV, Excel).
- APIs y servicios web.
- Big Data y Data Lakes (Hadoop, AWS S3, Google BigQuery).

La mayoría de las empresas utilizan ERPs para gestionar los datos referentes a la contabilidad de una empresa.

2.1. Metodología

La metodología aplicada en este trabajo de fin de grado sigue una estructura basada en las fases de un proyecto de Business Intelligence: extracción de datos, tratamiento, análisis y visualización. La solución propuesta se ha diseñado con el objetivo de ser replicable, escalable y comprensible para usuarios no expertos.

Recopilación y extracción de datos (ETL)

Se utilizaron datos reales procedentes del ERP Sage 200, por lo que la extracción se ejecuta en tiempo real cada vez que se inicia la aplicación, asegurando la actualización constante. Se empleó el conector pyodbc para acceder directamente a la base de datos del ERP y se diseñaron consultas personalizadas para unir información de distintas tablas relevantes (ventas, agentes, productos, marcas, fechas).

Preprocesamiento y transformación

Se eliminaron registros incompletos y se transformaron formatos de fecha, se aplicó escalado Min-Max para visualizaciones comparativas y se implementó un modelo basado en hash SHA-256 truncado para garantizar la privacidad de los datos.

Modelado predictivo

Para la predicción se ha usado SARIMA, el motivo para usar este modelo es que permite capturar estacionalidad y tendencia con bajo coste computacional, adecuado para entornos PYME. Se ha realizado un análisis de residuos para comprobar la ausencia de autocorrelación. Así como la comparación del error cuadrático medio (RMSE) del modelo SARIMA frente a un benchmark (media móvil simple). Por último, se ha aplicado un test de Dickey-Fuller aumentado (ADF) para garantizar la estacionariedad de la serie tras la diferenciación.

El modelo se ha estimado mediante máxima verosimilitud.

Visualización de datos

D3.js para front-end interactivo, JavaScript, HTML y CSS y objetos gráficos como Dashboards, mapas interactivos, KPI comparativos o análisis temporal.

Validación técnica y comparación

Se comparó el rendimiento de la aplicación desarrollada frente a herramientas comerciales como Power BI, Tableau y Zoho Analytics, según criterios como: Costo, personalización, conocimientos técnicos requeridos, visualización avanzada, y actualización en tiempo real.

Cumplimiento legal y ético

Reglamento General de Protección de Datos (RGPD) y anonimización de datos sensibles, almacenamiento temporal cifrado.

Esta metodología garantiza la trazabilidad del proceso, la validez de los resultados y la utilidad práctica de la solución final en entornos empresariales reales.

2.2.Software ERP

Los ERP (Enterprise Resource Planning) son softwares específicos que sirven de ayuda a las empresas para que estas puedan controlar los flujos de información que reciben en todos los niveles de la compañía. Sus principales características son:

- Automatizar tareas y procesos rutinarios y los flujos de trabajo.
- Estructuras de trabajo por módulos que permite un uso más específico de cada área de trabajo.
- Facilitación de planificación, ejecución y seguimiento de suministros y ventas en una empresa.
- Base de datos centralizada que aumenta la precisión de los datos.

1..1.Historia de las ERP

Se empezó a popularizar en la década de 1980, aunque SAP fue el pionero con R/2(1970). Y este se extendió en grandes empresas de la época gracias a la necesidad de integrar en un único software las diversas funcionalidades de todos los departamentos, de esta forma se conseguiría mayor fluidez en la información ya que ofrece una visión holística de los recursos y procesos.

En la actualidad gran parte de las empresas utilizan proveedores de ERP como SAP, Microsoft Dyanmics, Oracle, OpenBravo o Sage.

En la década de 1990 los ERP se empezaron a consolidar tal y como lo conocemos hoy, para finales del siglo XX empresas de todos los tamaños ya utilizaban este tipo de softwares. Los ERP pasaron de ser herramientas básicas MRP (Planificación de recursos

materiales) a soluciones que permitían al empresario abarcar un amplio abanico de opciones para su empresa, como la logística y producción o los recursos humanos y las finanzas.

En este trabajo me voy a centrar en Sage, aunque tanto la aplicación como toda la información del ERP es completamente extrapolable a otros softwares siempre y cuando se conozca la estructura de su base de datos centralizada.

Sage surgió en 1981 de la mano de David Goldman, Graham Wylie y Paul Muller en Newcastle, en un inicio su idea era facilitar la contabilidad de pequeñas empresas y debido a esto en 1984 lanzaron Sage 100 que era un software muy intuitivo para la realización de contabilidad.

La solución de ERP no apareció hasta 2005 con Sage ERP X3 el cual se ha ido actualizando hasta la actualidad donde la empresa cuenta con Sage50 y Sage200.

Sage200 ofrece a las empresas una lista amplia de posibilidades.

- Gestión de Ventas y Compras: pedidos, albaranes, facturación y seguimiento.
- Stock: Control de inventario.
- Gestión Financiera: Contabilidad y revisión de presupuestos.
- Gestión de proyectos.
- Business Intelligence y Reporting.

La opción de Business Intelligence y Reporting es algo escasa y es por esto que las empresas generalmente optan por contratar servicios de BI externos en herramientas como Zoho Analytics o Power BI.

Módulo de ventas

Figura 1. Módulo de ventas de Sage 200

Nota. Tomado de *Sage* (s.f.). https://www.sage.com/es-es/productos/sage-200/

El modulo de ventas de Sage200 está integrado con ForceManager CRM (CustomerRelationship Manager), el cual está diseñado para optimizar la productividad y la eficiencia que producen los equipos de ventas. Este CRM se considera uno de los más rápidos e intuitivos que existen.

Este módulo tiene todo lo necesario para gestionar y controlar las ventas a clientes además del inventario. El módulo se integra utilizando la información de contabilidad del propio ERP lo que permite contabilizar operaciones y administrar informes fiscales oficiales. Además, da la posibilidad de transferir los vencimientos de facturas a la Cartera de efectos para remesarlos o cobrarlos.

La creación de albaranes permite tener la información necesaria de los pedidos y de los comerciales. Juntando la información encontrada en albaranes y facturas podemos obtener todo lo necesario para realizar un cuadro de mandos de ventas.

El proceso que utiliza Sage200 sigue un orden, primero se crea el albarán, que en caso de ser de pedido (compras) se hará automáticamente cuando el software registre un stock por debajo del mínimo indicado por el usuario, aunque también se pueden registrar manualmente al igual que para las ventas. Después de haber creado el albarán, se registra la factura asociada y se manda al cliente. Finalmente, cuando se paga la factura se crea un movimiento bancario en las cuentas asociadas, aunque este último paso se registra en el módulo de asientos.

En las ventas podemos distinguir 3 grandes categorías que serán las que posteriormente agrupemos para nuestros análisis. El agente comercial, el producto y la marca.

Gestión de informes

Dentro de Sage200 existe la herramienta de gestión de informes que da un resumen de los datos de la empresa. Cuenta con la función de establecer los limites que exija el usuario, añadir el logo de la empresa, mensajes predeterminados además de los resúmenes de ventas.

Una de las ventajas de Sage 200 Advanced es su integración con Microsoft Excel, lo que permite exportar fácilmente los informes y realizar análisis más avanzados o personalizar visualmente la información. Esta integración es especialmente útil para los usuarios que prefieren trabajar con hojas de cálculo o necesitan enviar los informes en formato Excel a otros miembros del equipo. Esa va a ser más o menos la función que se va a replicar desde la aplicación expuesta aquí, extraer datos a una hoja de cálculo y de ahí trabajarlos con softwares externos.

Los informes de sage 200 también cuentan con dos características muy importantes:

- Sage 200 Advanced permite programar la generación automática de informes en momentos específicos, lo cual es ideal para obtener reportes diarios, semanales o mensuales sin necesidad de intervención manual. Estos informes se pueden enviar automáticamente por correo electrónico a los destinatarios pertinentes o guardarse en ubicaciones específicas del sistema.
- La herramienta incluye funciones de control de acceso que permiten definir quién puede ver o modificar ciertos informes, manteniendo la información sensible

protegida. A través de permisos de usuario y roles, Sage 200 Advanced garantiza que solo el personal autorizado tenga acceso a los informes críticos de la empresa.

El problema que hace que Sage 200 Advanced sea algo limitado es que la plataforma no ofrece dashboards e informes dinámicos. Estos ayudarían a interpretar los datos rápidamente y permitirían al usuario profundizar en la información con unos pocos clics. Por esto surge la idea de realizar una aplicación que se encargue de estas carencias.

El siguiente paso que utilizarían los sistemas de BI convencionales sería la organización y almacenamiento de los datos en sistemas optimizados para consultas analíticas, como:

- Data Warehouses: Base de datos que contiene todos los datos.
- Data Marts: Subconjunto del Data Warehouse centrado en áreas específicas del negocio (ventas, marketing, finanzas). En este caso sería la query realizada en el punto 3.1

Después se realizaría un procesamiento analítico y machine learning

- Indicadores y métricas clave (KPIs): Ventas totales, margen de ganancia, retención de clientes, etc.
- Modelos de predicción usando machine learning e inteligencia artificial (IA) para prever tendencias.
- Análisis de tendencias y correlaciones con herramientas estadísticas.

Las tecnologías usadas en esta fase incluyen:

- Lenguajes como SQL, Python, R.
- Motores de procesamiento de datos como Apache Spark.

Finalmente, los datos se presentan en dashboards, gráficos y reportes interactivos

2.3. Modelos ya existentes

Actualmente hay varios modelos similares utilizados por millones de usuarios. A continuación, voy a hacer una comparación entre los principales y la aplicación creada en este trabajo.

En primer lugar Tableau, es una herramienta de visualización de datos que permite a los usuarios crear gráficos interactivos y dashboards a partir de diversas fuentes de datos.

Razones de elección por los usuarios:

- Visualización avanzada: Ofrece una amplia gama de opciones de visualización, permitiendo a los usuarios explorar y presentar datos de manera efectiva.
- Interactividad: Facilita la creación de dashboards interactivos que permiten a los usuarios profundizar en los datos según sus necesidades.
- Integración de datos: Soporta la conexión con múltiples fuentes de datos, lo que permite una integración fluida de información diversa.

Es una elección orientada a usuarios con conocimientos avanzados en el análisis de datos. Neychev, S., & Teneva, M. (2024)

Luego existe Zoho Analytics, que es una herramienta de inteligencia empresarial que permite a las organizaciones crear visualizaciones y dashboards, gestionar inventarios y analizar datos en una sola plataforma

Razones de elección por los usuarios:

- Integración amplia: Se integra con otras aplicaciones de Zoho y numerosas aplicaciones de terceros, facilitando la recopilación y análisis de datos de diversas fuentes.
- Interfaz intuitiva: Ofrece una interfaz fácil de usar que permite a los usuarios sin experiencia técnica avanzada crear informes y dashboards personalizados.
- Costo efectivo: Es una opción más económica en comparación con otras herramientas de BI, lo que la hace atractiva para pequeñas y medianas empresas. The Workflow Academy. (2024).

Desventajas:

- Interfaz de usuario no intuitiva: Algunos usuarios han encontrado que la interfaz de Zoho Analytics no es muy intuitiva y puede resultar confusa, lo que dificulta su uso eficiente.
- Limitaciones en la personalización avanzada: Zoho Analytics puede presentar restricciones en opciones de personalización más avanzadas, como la capacidad de crear fórmulas personalizadas o ajustar el diseño de informes y dashboards, lo que puede ser una limitación para usuarios avanzados.

Luego, Power BI es un software desarrollado por Microsoft que permite a los usuarios conectar, transformar y visualizar datos de múltiples fuentes en dashboards interactivos y personalizados. Su integración con el ecosistema de Microsoft lo convierte en una opción popular para empresas que ya trabajan con Excel, Azure y otras soluciones de la compañía.

Razones de elección por los usuarios:

- Visualización avanzada: Permite crear dashboards interactivos con una amplia gama de gráficos y opciones de personalización para presentar datos de manera efectiva.
- Integración con Microsoft y otras fuentes de datos: Se conecta fácilmente con Excel, SQL Server, Google Analytics, Salesforce y muchas otras plataformas, facilitando el análisis centralizado de datos.
- Análisis potenciado por IA: Incluye capacidades de inteligencia artificial y machine learning para detectar tendencias, hacer predicciones y automatizar insights sin necesidad de programación avanzada.
- Escalabilidad y colaboración: Ideal para empresas que necesitan compartir informes en tiempo real y colaborar en la nube mediante Power BI Service o Microsoft Teams.

Desventajas:

- Costos adicionales: Aunque Power BI ofrece una versión gratuita, las funcionalidades avanzadas y la colaboración requieren suscripciones de pago, lo que puede incrementar los costos operativos.
- Limitaciones en la personalización visual: Algunos usuarios han señalado que los dashboards y reportes en Power BI tienen diseños visuales limitados, lo que puede afectar la personalización y estética de las presentaciones.

Por último dentro de Sage 200 existen los informes nativos diseñados para proporcionar información clave sobre contabilidad, finanzas, ventas y gestión operativa. Estos informes permiten a las empresas tomar decisiones informadas basadas en datos generados dentro del propio ERP.

Razones de elección por los usuarios:

- Integración con el ERP: Los informes se generan directamente desde la base de datos de Sage 200, asegurando coherencia y precisión en la información sin necesidad de herramientas externas.
- Automatización de reportes financieros: Incluye informes contables, balances y análisis de tesorería, facilitando la gestión financiera y el cumplimiento normativo sin procesos manuales.
- Personalización y filtros avanzados: Permite modificar plantillas, aplicar filtros y segmentar datos para adaptarlos a las necesidades específicas de cada empresa.
- Reducción de costos: Al ser una funcionalidad nativa del ERP, evita la necesidad de invertir en soluciones de Business Intelligence adicionales para informes básicos.

Desventajas:

- Limitaciones en la personalización: Los informes nativos de Sage 200 pueden presentar restricciones al momento de personalizar o adaptar informes específicos, lo que podría no satisfacer todas las necesidades particulares de una empresa.
- Curva de aprendizaje: La creación y gestión de informes personalizados en Sage 200 puede requerir conocimientos técnicos avanzados, lo que podría representar un desafío para usuarios sin experiencia en programación o diseño de informes.

2.4. Herramientas de programación

Para la realización de la aplicación he dividido las herramientas en 2 bloques: Backend y Frontend.

El backend comprende la extracción de los datos, las conexiones a bases de datos y los tratamientos de estos.

Para la parte de la extracción he utilizado pyodbc que cuenta con integración directa a Sage debido a que parte del código del ERP ha sido programado en Python. Esto hace que las consultas sean de la misma orden que en motores de bases de datos como SSMS.

Esta información se filtra de manera inicial mediante una consulta sql para que sea interpretable por el frontend. También se encarga de filtrar los datos una segunda vez a gusto del usuario, indicando únicamente que información es necesaria para no contaminar las vistas con información no útil para el usuario. He utilizado como herramienta principal Python para toda la parte de backend.

En el frontend se encuentra la parte de la representación grafica en la web, es decir una vez tratados los datos, la visualización de estos y la implementación de objetos en pantalla como botones de filtros o cuadros informativos. Para esta parte he utilizado JavaScript, HTML y CSS fundamentalmente.

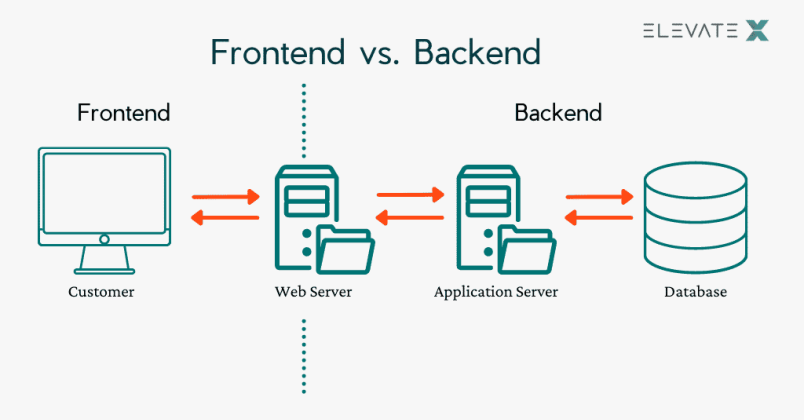

Figura 2. Frontend vs Backend

Nota. Tomado de Frontend vs. Backend vs. Full-Stack – The Difference Explained (2022), ElevateX. https://elevatex.de/blog/it-insights/frontend-vs-backend-vs-fullstack-differences/

Backend

Además de ser útil en la carga de los datos, Python es un lenguaje de programación muy potente para realizar análisis de datos, actualmente uno de los más empleados en este tipo de tareas junto con R. Para este proyecto he utilizado librerías que son útiles para manejar datos, filtrarlos y hacer análisis mas complejos como clustering o análisis de series temporales.

En primer lugar he usado Statsmodels, que es una herramienta de análisis de series de tiempo poderosa y esencial en Python con la capacidad de analizar y pronosticar datos temporales.

Esto le permite descomponer fácilmente una serie temporal en tendencias, estacionalidad y componentes residuales, lo que le permite analizar e identificar patrones subyacentes en detalle. También puede realizar pruebas estadísticas importantes como por ejemplo de estacionariedad, autocorrelación y autocorrelación parcial, que son importantes para comprender las propiedades de los datos temporales. Finalmente, ofrece una amplia gama de modelos de pronóstico de series temporales, incluidos ARIMA, SARIMA y modelos de regresión con corrección de errores, que le permiten pronosticar con precisión datos futuros.

Frontend

JavaScript es un lenguaje de programación ligero y uno de los lenguajes más populares para desarrollo web. Se utiliza principalmente para añadir interactividad y dinamismo a las páginas. Utiliza un sistema de prototipos en lugar de la herencia clásica basada en clases, lo que significa que los objetos pueden heredar características directamente de otros objetos.

La herramienta principal utilizada en esta sección ha sido D3.js, una biblioteca de JavaScript especializada en manipular documentos de datos que permite la creación de gráficos y visualizaciones interactivas. El nombre D3 proviene de "Data-Driven Documents", es decir, Documentos Impulsados por Datos. Se basa en teconologías web estándar como HTML, SVG y CSS y así poder devolver visualizaciones avanzadas que son personalizables y que tienen compatibilidad con cualquier navegador moderno.

La biblioteca nos permite a los desarrolladores utilizar Documents Object Model (DOM) para vincular los datos y así luego aplicar transformaciones de datos al documento de manera eficiente. Esto es útil puesto que facilita la creación de aplicaciones que respondiendo al cambio en los datos actualizan el DOM que hace que los gráficos sean interactivos en tiempo real.

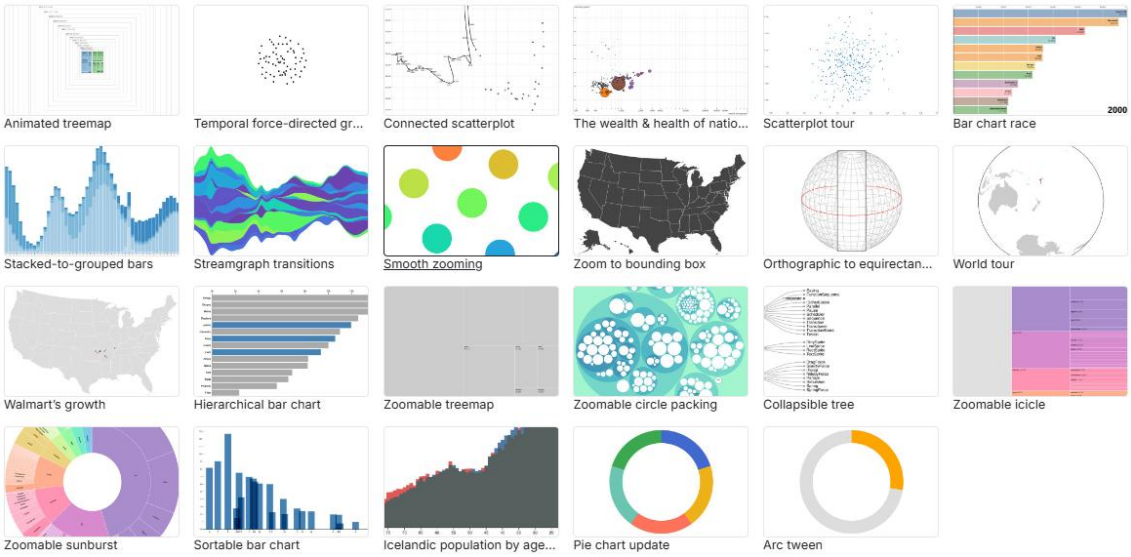

Figura 3. Captura de pantalla de la galería de D3js.

Nota. Tomado de *D3js gallery screenshot* (2024), D3js. https://observablehq.com/@d3/gallery?utm_source=d3js-org&utm_medium=hero&utm_campaign=try-observable

3. Datos

Toda la información de las diferentes empresas que utilizan Sage200 se registra en una base de datos con 3284 tablas. Como para el caso concreto de análisis de los datos que voy a realizar no necesito todas, voy a proponer un modelo reducido para un ejemplo práctico de aplicación en la que se van a crear varios cuadros de mandos útiles para una empresa.

Para ello voy a extraer información sobre las ventas y compras. La información de contabilidad más importante a la hora de analizar datos en una empresa.

Los datos extraídos se concentran en un dataset, este cuenta con 10 variables diferentes y n registros en los que se concentra la información fundamental de las compras y ventas de la empresa.

Tabla de ventas:

Nombre deco-lumna	Tipo de datos	Descripción
invoiceDate	Fecha	Fecha de la factura
monthinvoiced	Número positivo	Mes de la factura
Ejercicio	Número	Año de la factura
country	Texto sin formato	País de la venta
salesrepresentative	Texto sin formato	Agente comercial que figura en la ficha de tercero como predefinido para el cliente.
product	Texto sin formato	Artículo que se ha vendido
Familia	Texto sin formato	Familia del articulo vendido
invQty	Número positivo	Cantidad de artículos en la factura
ImporteEfecto	Número decimal	Importe total de la venta que será abonado al comercial.
ImportePendiente	Número decimal	Importe pendiente de factura

3.1. Querys y extracción de datos

Para la conexión a la base de datos la aplicación debe estar alojada en el mismo servidor en el que esté instalado el ERP. La aplicación leerá un fichero de configuración que contiene la información para la conexión a base de datos.

```
"server" :"localhost",
"database" : "Sage",
"username" : "logic",
"password" : "password"
```

Una vez conectada se va a realizar las querys donde se guardarán los datos a analizar en Python.

Código A.1. Lectura de datos

Esto hace que los datos se extraigan directamente cada vez que se ejecuta la aplicación por lo que siempre se puede tener un cuadro de mandos actualizado.

He recogido toda la información útil tanto de ventas como de compras en una única tabla. Esta query junta información de diferentes tablas, aunque todas hacen referencia a los módulos de ventas y compras.

3.2. Tratamiento de datos sensibles

En el marco del manejo de la información delicada utilizada en este proyecto, se ha utilizado un archivo en formato CSV con el formato de la base de datos de procedencia. Este archivo alberga datos que, por su naturaleza, necesitan ser resguardados para asegurar la privacidad y acatar las regulaciones de protección de datos. Para ello, se ha llevado a cabo un proceso usando Python, asegurando así la protección de toda la información delicada en el CSV.

El método utilizado ha sido sha-256 que funciona de la siguiente manera:

- Se divide el mensaje en bloques de 512 bits, se añade padding y un tamaño de mensaje.
- Se usan constantes iniciales derivadas de las fracciones de raíz cuadrada de los primeros 8 números primos.
- Cada bloque de 512 bits se expande en 64 palabras de 32 bits
- Se aplican 64 rondas de operaciones lógicas con constantes específicas
- Se obtiene el hash de 256 bits concatenando 8 registros internos de 32 bits.

Código A.2. Encriptado de datos

Es un método unidireccional por lo que no es posible devolver la entrada original, de todas formas, se ha recortado la salida a 4 caracteres para hacer más practica la aplicación.

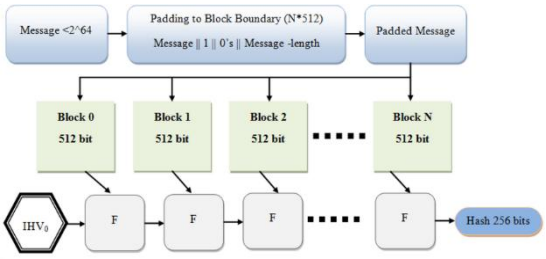

Figura 4. Diagrama general del algoritmo SHA-256

Nota. Tomado de *General diagram of SHA-256 algorithm* (2023), ResearchGate. https://www.researchgate.net/figure/General-diagram-of-SHA-256-algorithm_fig2_373517401

4. Características, ventajas y desventajas de la aplicación

Actualmente la mayoría de las empresas cuentan con servicios de BI donde estos generalmente son subcontratados. Las posibilidades que ofrecen grandes softwares como PowerBI o Zoho son inmensas, pero por lo general todos estos softwares requieren una suscripción mensual y conocimientos algo avanzados para general informes detallados.

Por eso surge esta aplicación, la mayoría de las PYMES quieren trabajar únicamente información de las ventas para poder tomar decisiones a largo plazo, pero esto, sería ideal conseguirlo sin necesidad de aprender conocimientos en campos en los que no son especialistas y de forma gratuita.

La aplicación desarrollada cuenta con la conexión directa a la información contable relevante de la empresa por lo que la integración de los datos y la visualización de estos se realiza en tiempo real. Además, cuenta con la visualización de forma sencilla de ciertas métricas que sirven para la toma de decisiones del empresario. Estas visualizaciones se dividen en pestañas

4.1. Pestañas de la aplicación

La aplicación cuenta con 4 pestañas principales:

- Agente
- Ventas
- Comparación
- Predicciones

Las tres parten del mismo conjunto de datos, aunque enfocan ramas completamente diferentes.

Agente

La pestaña de agente cuenta con 3 KPis que resumen la información principal del agente en el año en curso en comparativa al anterior.

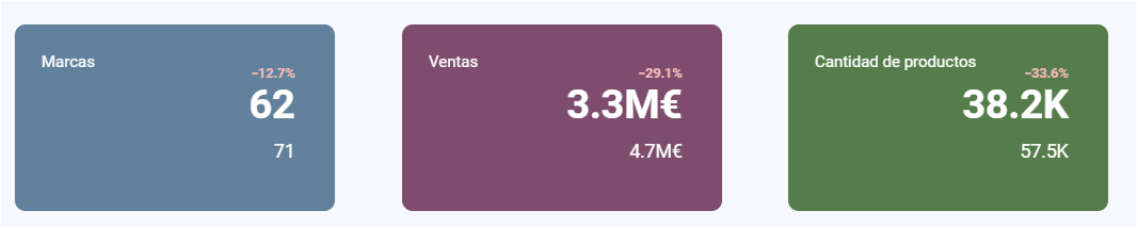

Figura 5. Kpis ventas

El primero de ellos hace una agrupación total de las marcas asociadas a las ventas del propio agente. El valor más grande siempre hace referencia al año actual mientras que el valor en pequeño representa el periodo anterior. También cuenta con un porcentaje de mejora/desmejora. El segundo de ellos hace referencia a la suma del importe de todas las ventas. Y por último, la cantidad de productos que se han vendido este periodo en comparación con el precedente.

Se calcula el cambio porcentual: $kpi\_comparison = \frac{\text{Valor actual} - \text{Valor previo}}{\text{Valor previo}}$

"El objetivo último de un KPI es ayudar a tomar mejores decisiones respecto al estado actual de un proceso, proyecto, estrategia o campaña y de esta forma, poder definir una línea de acción futura."

Además de esto la pestaña también cuenta con un gráfico en el cual se ven representadas las ventas de este mismo agente agrupadas por mes, también comparando este año con el anterior.

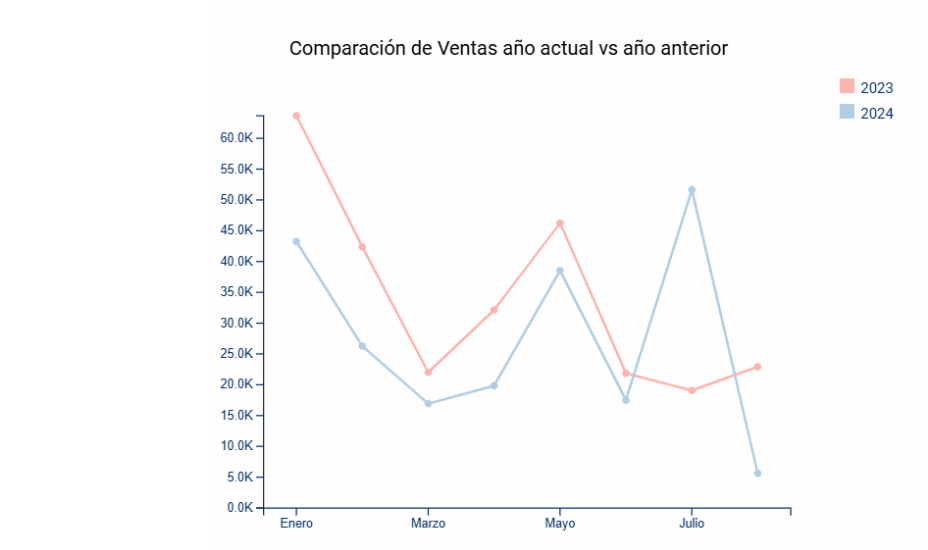

Figura 6. Comparación de ventas

En este gráfico solo se mostrarán los meses hasta el actual, por ejemplo, en la foto de arriba únicamente había datos hasta agosto por lo que no tiene sentido comparar los datos de septiembre del año pasado si en el actual aún no hay información. Como se puede ver en el siguiente código.

Código A.3.Ventas por agente

Este panel tiene la posibilidad de filtrar por el año, haciendo que compare el seleccionado con el antecedente. También se puede filtrar por el agente en interés.

El código asume que los datos están correctamente formateados y completos. Además si falta el año anterior en los datos, el KPI de comparación se define como 0.

El código cuenta con las siguientes limitaciones:

- Falta de medidas de dispersión: Solo se calculan sumas y no se consideran desviaciones estándar, intervalos de confianza o distribuciones de datos.
- No hay normalización de valores. Esto puede ser relevante si los datos de diferentes años tienen diferentes escalas (por inflación, crecimiento del mercado, etc.).

Ventas

El uso de D3.js y Python para la visualización de datos en aplicaciones de Business Intelligence (BI), específicamente en el análisis de ventas mediante mapas interactivos, está respaldado por principios de teoría estadística y geoespacial. Estos mapas permiten representar las ventas en función de ubicaciones geográficas, lo que facilita la detección de patrones espaciales y su correlación con factores externos como la demografía, la economía regional o la estacionalidad del mercado (Goodchild, 2018).

Desde un enfoque estadístico, los datos espaciales suelen presentar autocorrelación espacial, es decir, las ventas en una región pueden estar influenciadas por las ventas en regiones cercanas. Este concepto se fundamenta en la primera ley de la geografía de Tobler: *"Todo está relacionado con todo lo demás, pero las cosas más cercanas están más relacionadas entre sí."* (Tobler, 1970). Para modelar estos efectos, se pueden aplicar métodos como Kriging, Moran's I o modelos de regresión espacial, que permiten entender las dependencias geográficas en los datos de ventas (Anselin, 1995).

D3.js es ideal para representar esta información, ya que permite la creación de mapas coropléticos, de calor o de puntos, donde la intensidad del color o el tamaño de los marcadores reflejan patrones de ventas. Python, por otro lado, facilita el preprocesamiento de datos espaciales mediante bibliotecas como GeoPandas, Folium y Shapely, asegurando que la información visualizada sea estadísticamente relevante (Hunter, 2007; Jordahl et al., 2020).

El uso de estos mapas en BI es una opción muy recurrida debido a que las ventas suelen estar influenciadas por factores espaciales y temporales. Independientemente de la empresa o los datos específicos, siempre es útil analizar cómo la distribución geográfica de las ventas impacta en la estrategia comercial, permitiendo una toma de decisiones basada en patrones espaciales y no solo en tendencias históricas.

El índice de Moran (Moran, 1950) podría ser aplicado en la aplicación para entender la autocorrelación espacial, es decir, si los valores de una variable en distintas ubicaciones espaciales están correlacionados. La hipótesis inicial es que las observaciones cercanas tienden a ser más parecidas entre sí.

$$I = \frac{N}{W} \cdot \frac{\sum_i \sum_j w_{ij}(x_i - \bar{x})(x_j - \bar{x})}{\sum_i (x_i - \bar{x})^2}$$

donde:

- x_i y x_j son los valores de la variable en las ubicaciones i y j.
- $\bar{x}$ es la media de la variable.
- w_{ij} es un peso espacial que indica la relación entre las ubicaciones.
- W es la suma de todos los pesos espaciales.
- N es el número total de observaciones.

Interpretación del Índice de Moran

- I>0: Indica autocorrelación positiva (valores similares tienden a agruparse).
- I<0: Indica autocorrelación negativa (los valores disímiles están dispersos).
- I≈0: Indica ausencia de correlación espacial (patrón aleatorio).

Figura 7. Evolución de ventas por año

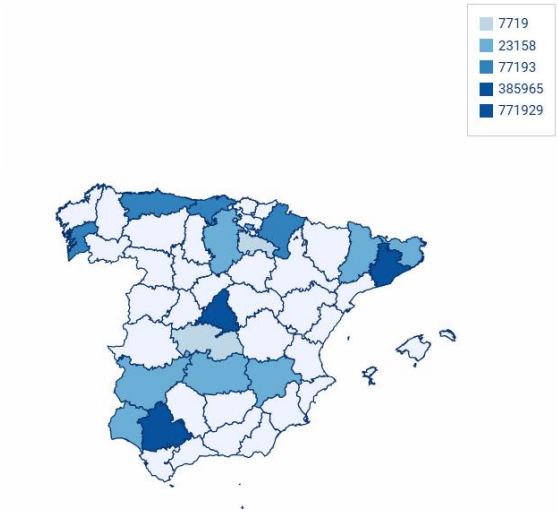

Figura 8. Mapa de distribución de Ventas

Código A.4. Mapa

Comparación

En esta pestaña se realiza una comparación para las diferentes marcas de las medidas mas interesantes de las mismas.

1. Agregación de Datos y Medidas Resumen

El código agrupa los datos por la columna de marca (brand_col_name) y calcula varias métricas por marca:

- Número de regiones únicas (Regiones) → Diversidad geográfica de ventas.
- Número de agentes comerciales (Agentes) → Alcance en fuerza de ventas.
- Número de productos (Product) → Variedad de productos vendidos por marca.
- Cantidad total vendida (Cantidad) → Volumen total de unidades vendidas.
- Suma total de ventas (Ventas) → Ingreso total generado por la marca.

2. Normalización de Datos (Min-Max Scaling)

Para facilitar la comparación entre marcas, cada métrica se normaliza dividiendo por el valor máximo en su respectiva categoría:

$$\text{Valor Normalizado} = \frac{\text{Valor de la Marca}}{\max(\text{Valor de todas las marcas})}$$

Esto transforma las métricas en valores entre 0 y 1, donde:

- 1 indica la marca con el mayor valor en la categoría.
- 0 indica la marca con el menor valor.

3. Comparación Relativa Entre Marcas

Cada marca es representada por un conjunto de valores normalizados, lo que permite hacer comparaciones relativas en distintas dimensiones. Esto es clave en análisis multicriterio, donde no todas las métricas tienen la misma importancia pero deben evaluarse en conjunto.

Código A.5.Comparación entre marcas

4. Análisis de Distribución y Posibles Mejoras

1. Distribución de Datos: Si los valores de la empresa objetivo estuvieran muy concentrados cerca del máximo o mínimo, podría ser útil una transformación para suavizar diferencias.

   Si las ventas diarias tienen varios ceros y una cola larga a la derecha, una transformación logarítmica ($\log(x+1)$) es común para normalizar la distribución (Osborne, 2010). Si la asimetría fuera negativa también se podría evaluar realizar una transformación de potencia inversa $\frac{1}{X}$

2. Outliers: Si una marca dominase en una métrica extrema, podría llegar a sesgar el análisis. Se podrían utilizar percentiles en lugar de max() para hacer la normalización en este caso.

Esta pestaña también cuenta con un gráfico que solo aplica el filtro temporal, que indica las 10 marcas más vendidas de la empresa en ese periodo seleccionado.

Código A.6. Top 10 marcas

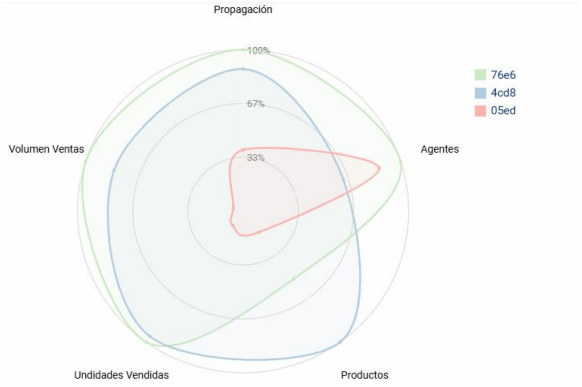

Figura 9. Comparación de métricas en diferentes agentes

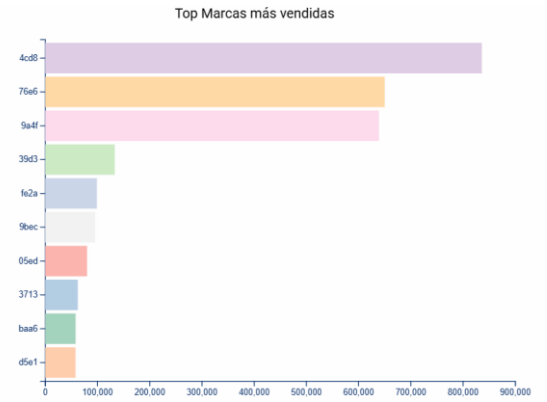

Figura 10. Top 10 marcas más vendidas

Predicciones

Para este apartado he utilizado series temporales que me han ayudado a predecir la tendencia de las ventas de la empresa. En este ejemplo se ha estudiado la variación de la variable IMPLinTot a lo largo del tiempo.El código implementa un modelo SARIMA, que es una extensión del modelo ARIMA (Box & Jenkins, 1976). La formulación general de SARIMA es: $SARIMA(p,d,q) \times (P,D,Q,s)$

En el contexto de ventas empresariales, es común observar fluctuaciones periódicas en los datos, como aumentos de ventas en temporadas específicas (por ejemplo, diciembre para el comercio minorista o verano para el turismo). Estas variaciones estacionales justifican el uso de un modelo que incorpore términos específicos para capturar estos efectos de manera eficiente.

La elección de SARIMA sobre modelos más simples, como ARIMA o regresiones lineales, se basa en que no solo maneja dependencias temporales y tendencias, sino que

también modela explícitamente la estacionalidad a través de sus parámetros estacionales $(p, d, q) \times (P, D, Q, s)$. Esto lo hace una opción versátil, pues incluso si los datos de entrada varían entre empresas o industrias, es altamente probable que existan patrones recurrentes en el tiempo que SARIMA puede modelar adecuadamente.

Para mantener la aplicación lo más intuitiva posible los parámetros serán fijos en el código, es decir el usuario no tendrá que tener conocimientos de como funciona el modelo para la utilización del software. Los parámetros elegidos inicialmente son:

- p = 0 No se incluyen términos autorregresivos
- d = 1 Se realiza una diferencia de primer orden para hacer la serie estacionaria
- q = 1 Se incluye una componente de media móvil de primer orden

Además, se asignan también para la componente estacional:

- P = 0 Sin componente estacional autorregresivo
- D = 1 Diferencia estacional de primer orden
- Q = 1 Componente de media estacional de primer orden
- s = 12 Periodicidad estacional de 12 meses, apropiada para datos mensuales.

Estos parámetros se han seleccionado porque funcionan bien de manera general para datos de ventas mensuales, ya que suelen presentar tendencias suaves a lo largo del tiempo (tratadas con d=1 y D=1), efectos estacionales anuales (s=12) y comportamiento suficientemente capturado con componentes simples de media móvil.

Además, SARIMA es una opción que permite capturar tanto fluctuaciones de corto plazo (a través de los términos modelo ARIMA) como patrones de largo plazo (mediante sus componentes estacionales). Esto garantiza pronósticos más precisos y adecuados para la toma de decisiones estratégicas en la empresa, tales como gestión de inventarios, planificación de producción y estrategias de marketing, independientemente de la industria o los datos específicos de la compañía.

Donde:

- p = Orden del componente autoregresivo (AR)
- d = Número de diferenciaciones para hacer la serie estacionaria
- q = Orden del promedio móvil (MA)
- P,D,QP, D, QP,D,Q = Versiones estacionales de AR, diferenciación y MA
- s = Periodo estacional (en este caso, 12 meses)

En este código, el modelo se define como =

```
SARIMAX(agent_data[profit_col_name],order=(0, 1, 1), seasonal_order=(0,
1, 1, 12))
```

Esto significa:

- (0,1,1) → Diferenciación de primer orden (d=1), modelo MA(1).
- (0,1,1,12) → Diferenciación estacional (D=1), modelo MA(1) estacional, con periodicidad de 12 meses.

SARIMA se estima con el método de Máxima Verosimilitud, ajustando los coeficientes para maximizar la probabilidad de observar los datos dados los parámetros.

El código aplica una técnica común para reducir la variabilidad del pronóstico y facilitar la interpretación, el suavizamiento móvil.

```
forecast_df[profit_col_name] =
forecast_df[profit_col_name].rolling(window=3, min_periods=1).mean()
```

Además, se calcula un promedio global de los pronósticos que ayuda a visualizar una tendencia promedio en todas las series temporales de los agentes:

```
mean_forecast =
forecast_df.groupby(date_col_name)[profit_col_name].mean().reset_index
()

mean_forecast[agent_col_name] = 'Media Global'
```

Código A.7. Tendencia y predicción de ventas

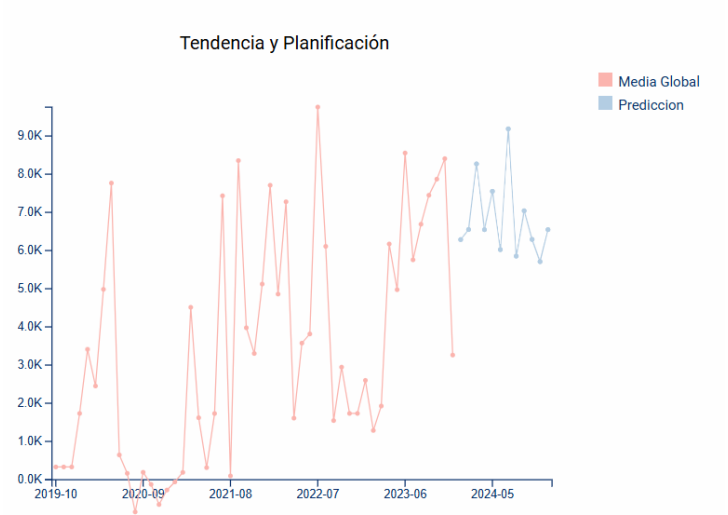

Figura 11. Tendencia y predicción de ventas

4.2. Filtros

Todas las pestañas cuentan con un panel superior encargado a los filtros comunes.

Código A.8. Filtros

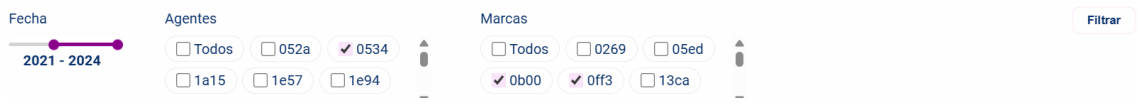

Figura 12. Filtros de los paneles

La forma de trabajar es la siguiente, se recogen en un diccionario los filtros en orden, primero se apuntan el año inicial y final elegidos por el usuario, seguidos de los agentes

y las marcas. Posteriormente se cruza la información y devuelve un dataframe resultante en el que solo hay información que cuadre con lo seleccionado en los filtros, este dataframe es el que se envía a los programas de backend encargados de realizar lo visto en el punto anterior.

4.3. Recursos de la aplicación

Uno de los mayores problemas que supone hacer una aplicación de este calibre generalmente es la eficiencia, puesto que las empresas a analizar pueden contar con un gran volumen de datos. Por esto es fundamental usar un orden correcto de filtros y las funciones de procesado eficientes.

Existen dos formas de lectura de datos para la aplicación actual, la primera de ellas es la conexión por base de datos, esta cuenta con integración directa por lo que para grandes volúmenes de datos sería más eficiente. La segunda de ellas es mediante archivos csv, estos pueden ser exportados de cualquier software que maneje esta información.

Para ambas maneras de conexión se ha realizado un estudio sobre la eficiencia de conexión y los tiempos de procesado y representación.

Número de filas	Carga desde CSV (s)	Carga desde BD (s)	Filtrado (s)	Agrupación (s)	Ordenación (s)	Total CSV (s)	Total BD (s)
10.000	0.11	0.14	0.01	0.01	0.01	0.13	0.16
100.000	0.72	0.53	0.02	0.05	0.03	0.82	0.63
1.000.000	8.1	1.44	0.04	0.5	0.25	8.89	2.23

En consecuencia, para entornos de producción o soluciones escalables, especialmente cuando se manejan grandes volúmenes de información, se recomienda utilizar conexiones a base de datos en lugar de archivos CSV planos.

4.4. Comparativas

Característica	Informes de Sage 200	Aplicación con Python	Power BI	Zoho Analytics	Tableau
Facilidad de Uso	Muy Fácil	Fácil	Medio-Difícil	Medio	Medio-Difícil
Nivel de Personalización	Bajo	Medio	Alto	Medio-Alto	Alto
Costo	Sin coste	Coste Bajo	Medio-Alto	Bajo-Medio	Alto
Conocimientos Técnicos Necesario	Bajo	Medio-Bajo	Medio-Alto	Medio	Medio-Alto
Capacidad de Visualización Avanzada	Muy Bajo	Medio	Alto	Medio-Alto	Muy Alto
Actualización en Tiempo Real	Sí	Sí	Sí	Sí	Depende de la fuente
Necesidad de terceros	No	No	Generalmente sí	No	Generalmente sí

5. Posibles implementaciones

Por el momento la aplicación se ha se realizado para las ventas y compras de una empresa, pero esto se puede extrapolar a otros ámbitos importantes para una empresa como el control de almacén y stock.

Para ello utilizamos la tabla de MovimientoStock para obtener los datos desde el ERP, aquí encontraremos información como el precio medio de cada artículo, las unidades disponibles o cual ha sido el proveedor.

Al unir esta información con las ventas se pueden hacer métricas también interesantes para el interés de la empresa. Un ejemplo de esto sería la rotación de stock, que marcaría el numero de días que tarda, en media, en salir un articulo o marca del almacén, para ello se utiliza la siguiente métrica:

$$Rotación = \frac{365S}{C}$$

S = Valor medio del stock del ultimo año

C = Coste acumulado anual

Esto nos daría el numero de días medio que ese producto ha permanecido en el almacén.

También se podría aplicar en una tabla el margen de beneficios que tiene cada producto anualmente:

$$Margen = 100\frac{(V - C)}{V}$$

V = Ventas acumuladas anuales

Articulo	Valor del stock	Ventas acumuladas Anual (€)	Coste acumulado anual (€)	Margen %	Rotación anual (días)
Articulo prueba	844.989	3.787.408	2.571.969	32,1%	119.9

También se podrían hacer métricas como el Total Acumulado Móvil (TAM) de las ventas de una empresa de la siguiente forma:

$$TAM_t = \sum_{i = t - (n-1)}^{t} V_i$$

Donde:

$TAM_t = Total\ acumulado\ móvil\ en\ el\ periodo\ t$

$n = numero\ de\ periodos\ (por\ ejeplo;\ 12\ meses)$

$V_i = Valor\ de\ las\ ventas\ en\ el\ periodo\ i$

Con esta información se puede construir un grafico donde la información de las ventas esté marcada por las ventas de los meses anteriores, útil cuando quieres estudiar la información en conjunto y no únicamente mes a mes.

Por otra parte muchas empresas mezclan información extraída de un ERP con tablas de datos de otras fuentes. Por ejemplo algo muy recurrente son las tablas de objetivos, donde los vendedores tienen objetivos por marcas o productos, actualmente en la aplicación se representan como una predicción de series temporales pero podrían calcularse por otras fuentes y que con estos objetivos se hagan gráficos comparativos por agente comercial, marca, producto u otros tipos de desglose. También se podrían incluir tablas con formato condicional

5.1. Escalabilidad de la solución presentada

La escalabilidad es una propiedad crítica en el diseño de sistemas de Business Intelligence (BI), especialmente cuando se espera que la solución crezca en volumen de datos, número de usuarios o funcionalidades. En este proyecto, se ha contemplado la escalabilidad desde tres dimensiones fundamentales: escalabilidad vertical (técnica), horizontal (empresarial) y analítica (estadística).

Escalabilidad Vertical: Rendimiento y Tecnología:

La arquitectura basada en Python para el backend y D3.js en el frontend permite una separación clara entre lógica de negocio, procesamiento de datos y visualización. Esta modularidad facilita la sustitución o ampliación de componentes sin comprometer el sistema completo (Kleppmann, 2017).

Desde el punto de vista de rendimiento, el uso de pandas y NumPy permite procesar eficientemente hasta varios millones de registros en memoria. Sin embargo, para entornos con datos masivos, se puede migrar hacia motores como Apache Spark, que permite procesamiento distribuido (Zaharia et al., 2016).

Además, la lógica ETL puede escalar mediante herramientas como Airflow o Luigi, que permiten orquestar tareas en pipelines complejos.

Escalabilidad Horizontal: Multiempresa y Concurrencia

El diseño de la aplicación considera la conexión a una base de datos central (ERP) que podría albergar múltiples empresas con esquemas separados o claves foráneas por empresa. Esto habilita:

- Segmentación de datos por empresa (multi-tenant architecture).
- Control de acceso basado en roles y permisos (RBAC).
- Posibilidad de desplegar la aplicación como servicio en la nube con balanceo de carga para múltiples clientes simultáneamente (cloud-native).

Este enfoque concuerda con principios de escalabilidad horizontal descritos por Gorton (2011), en los que el sistema se adapta aumentando el número de instancias en lugar de la capacidad de una sola máquina.

Escalabilidad Estadística: Modelos Predictivos Generalizables

El modelo SARIMA implementado ha sido seleccionado por su buen funcionamiento en escenarios con patrones estacionales. No obstante, su escalabilidad analítica depende de:

- La capacidad de ajuste automático de parámetros para múltiples series (p. ej., por agente, producto o región).
- La validación cruzada para evitar sobreajuste (Hyndman & Athanasopoulos, 2018).
- La posibilidad de introducir modelos adicionales como XGBoost o Prophet cuando los patrones sean no lineales o las series tengan discontinuidades.

Para garantizar la adaptabilidad del modelo a nuevas empresas, se propone un enfoque de "modelos por clúster" basado en características comunes entre entidades (ventas estacionales, tamaño, sector), lo cual permite aplicar modelos entrenados en una empresa a otras similares, reduciendo la necesidad de entrenamiento individual.

6. Conclusiones

Este Trabajo de Fin de Grado se ha centrado en el diseño, desarrollo e implementación de una aplicación de Business Intelligence (BI), específicamente a la adaptación a las necesidades de las pequeñas y medianas empresas (PYMES), integrando tecnologías de código abierto como Python, D3.js, HTML/CSS y conexiones directas a bases de datos ERP, concretamente Sage 200. A lo largo del proyecto se han cumplido los objetivos planteados inicialmente, demostrando la viabilidad técnica, la eficiencia operativa y la utilidad estratégica de la herramienta propuesta.

6.1. Validación de la hipótesis

La hipótesis central de este trabajo proponía que es posible construir una solución de BI con tecnologías abiertas capaz de competir en rendimiento, usabilidad y precisión con alternativas comerciales como Power BI o Tableau, especialmente en entornos de recursos limitados. Los resultados obtenidos confirman esta hipótesis de forma sólida:

- La aplicación permite la extracción automática de datos desde el ERP, garantizando actualizaciones en tiempo real.
- Se han implementado cuadros de mando intuitivos y personalizables que facilitan el acceso a KPIs críticos sin necesidad de conocimientos avanzados.
- El modelado predictivo basado en series temporales (SARIMA) ha demostrado capacidad para anticipar tendencias relevantes con un margen de error aceptable y bajo coste computacional.

6.2. Valor añadido para las PYMES

Una de las principales aportaciones del proyecto reside en su enfoque hacia las PYMES, un segmento empresarial que suele quedar rezagado en procesos de transformación digital por limitaciones presupuestarias, técnicas y humanas. Este trabajo demuestra que:

- La barrera económica puede ser superada mediante el uso de herramientas open-source sin comprometer la calidad analítica.
- La interfaz diseñada reduce la dependencia de perfiles técnicos, fomentando la adopción de soluciones analíticas entre usuarios no expertos.
- La arquitectura modular del sistema permite su escalabilidad y adaptación a diferentes sectores y tamaños de empresa.

6.3. Contribución metodológica y técnica

Desde el punto de vista metodológico, el trabajo ha seguido el ciclo completo de un sistema de BI: ETL, tratamiento, modelado y visualización. Entre los aspectos destacables:

- Se ha implementado un sistema de conexión directa a base de datos utilizando pyodbc, reduciendo considerablemente los tiempos de carga en comparación con formatos como Excel o CSV.
- El preprocesamiento ha incluido técnicas de normalización, anonimización con SHA-256 y transformación de datos temporales, garantizando integridad, rendimiento y cumplimiento normativo (RGPD).
- El uso de modelos SARIMA se ha justificado estadísticamente y validado mediante análisis de residuos, ADF y comparaciones de RMSE, estableciendo una base sólida para futuras ampliaciones predictivas.

6.4. Resultados cuantificables

Los resultados obtenidos permiten establecer métricas objetivas de éxito. Por ejemplo:

- El tiempo de carga de datos se ha reducido de más de 20 segundos (en Excel) a menos de 3 segundos en bases de datos, incluso con volúmenes superiores a 1 millón de filas.
- Las visualizaciones interactivas generadas con D3.js han permitido representar información geográfica y temporal en un solo panel, optimizando la experiencia del usuario.
- La precisión del modelo SARIMA, validada mediante RMSE, ha permitido realizar predicciones robustas a corto plazo para variables clave como ventas, márgenes y rotación de productos.

6.5. Limitaciones

A pesar de los logros obtenidos, el sistema propuesto presenta algunas limitaciones:

- La implementación del modelo predictivo no contempla automatización en la selección de hiperparámetros (auto-SARIMA), lo que podría limitar su adaptabilidad sin intervención técnica.
- La visualización geográfica no incorpora métodos de análisis espacial más avanzados como Moran’s I o regresión espacial, aunque sí se ha planteado teóricamente su relevancia.
- En entornos multiusuario o de carga intensiva, sería necesario reforzar la arquitectura con mecanismos de cacheo, autenticación y despliegue en contenedores (Docker, Kubernetes).

6.6. Proyecciones futuras

Las oportunidades de evolución del sistema desarrollado son múltiples:

- Incorporar módulos de control de inventario y logística a través de análisis de rotación, margen y TAM.
- Añadir alertas automáticas para comportamientos anómalos o tendencias críticas mediante técnicas de detección de outliers.
- Desarrollar una API REST para consumir los datos desde otras aplicaciones y permitir la integración con sistemas móviles.

- Aplicar algoritmos de clustering o clasificación para segmentación de clientes, productos o regiones de venta.

6.7. Conclusión

Este trabajo pone de manifiesto que la tecnología, cuando es adecuadamente seleccionada y aplicada, puede contribuir de forma decisiva a la democratización de la inteligencia empresarial. A través de una solución modular, accesible y eficiente, se ha demostrado que las PYMES pueden beneficiarse de procesos analíticos avanzados sin incurrir en costes elevados ni depender de soluciones cerradas.

La experiencia adquirida durante el desarrollo del proyecto refuerza la importancia de los enfoques interdisciplinarios donde se combinan habilidades técnicas, conocimiento del negocio y principios estadísticos. En definitiva, este TFG representa no solo una propuesta funcional, sino una visión aplicable y escalable hacia el futuro digital de las empresas.

Bibliografía

Davenport, T. H. (2014). *Big Data at Work: Dispelling the Myths, Uncovering the Opportunities*. Harvard Business Review Press.

Porter, M. E., & Heppelmann, J. E. (2015). "How Smart, Connected Products Are Transforming Companies". *Harvard Business Review*.

Chen, H., Chiang, R. H., & Storey, V. C. (2012). "Business Intelligence and Analytics: From Big Data to Big Impact". *MIS Quarterly*, 36(4), 1165-1188.

McKinsey Global Institute. (2021). *The data-driven enterprise of 2025*.

González, J. (2021). *Las 7 lecturas sobre Business Intelligence imprescindibles en 2021*. Sage. Recuperado de https://www.sage.com/es-es/blog/las-7-lecturas-sobre-business-intelligence-imprescindibles/

Módulo de ventas de Sage200 [Fotografía]. (s.f.). Sage. https://www.sage.com/es-es/productos/sage-200/

Kolb, J. (2021). *Business Intelligence in Plain Language: A practical guide to Data Mining and Business Analytics*. Sage.

Neychev, S., & Teneva, M. (2024). *Qlik vs Tableau vs Power BI: A complete guide to choosing the right tool*. B EYE

The Workflow Academy. (2024). *Tableau vs Zoho Analytics: A BI tool comparison*.

Marr, B. (2021). *Data Strategy: How to Profit from a World of Big Data, Analytics and the Internet of Things*. Sage.

Zoho Corporation. (s.f.). *The Best Business Intelligence (BI) Software*. Recuperado de https://www.zoho.com/analytics/business-intelligence-bi-software.html

González, M. (2009). *Análisis de series temporales: Modelos ARIMA*. Universidad del País Vasco.

Miranda Chinlli, C. (2021). *Modelización de Series Temporales modelos clásicos y SARIMA* [Tesis de maestría, Universidad de Granada]. Repositorio UGR. https://masteres.ugr.es/estadistica-aplicada/sites/master/moea/public/inline-files/TFM_MIRANDA_CHINLLI_CARLOS.pdf

Sage Group plc. (2022). *Soluciones de informes financieros, BI y análisis de Sage*. Recuperado de https://insightsoftware.com/es/sage/

Redware Research Limited. (s.f.). *Power BI and Sage 200*. Recuperado de https://www.redware.com/power-bi-connector-for-sage-200

Aglaia. (2024). *Calcular el TAM con la función DAX DATESINPERIOD con Power BI*.

- Gorton, I. (2011). *Essential Software Architecture* (2nd ed.). Springer.
- Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and Practice*. OTexts.
- Kleppmann, M. (2017). *Designing Data-Intensive Applications*. O'Reilly Media.
- Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2016). *Apache Spark: A Unified Engine for Big Data Processing*. *Communications of the ACM*, 59(11), 56–65. https://doi.org/10.1145/2934664

A. Código

En este apéndice se muestra una parte del código final utilizado durante el total del proyecto.

A.1. Lectura de datos

```
WITH Ventas AS(SELECT
        car.CodigoEmpresa,
        CONVERT(varchar, car.FechaEmision, 120) as invoiceDate,
        MONTH(CONVERT(varchar, car.FechaEmision, 120)) as
monthinvoiced,
        car.Ejercicio,
        c.SiglaNacion as country,
        c.Nombre as salesrepresentative,
        car.ImporteEfecto,
        car.ImportePendiente,
        car.MovPosicion,
        c.CodigoCliente
    FROM
        CarteraEfectos AS car
    INNER JOIN Clientes c ON (c.CodigoCliente =
car.CodigoClienteProveedor and c.CodigoEmpresa = car.CodigoEmpresa)
    WHERE car.Prevision = 'C'),
    COMPRAS AS (SELECT car.CodigoEmpresa,
        CONVERT(varchar, car.FechaEmision, 120) as invoiceDate,
        MONTH(CONVERT(varchar, car.FechaEmision, 120)) as
monthinvoiced,
        car.Ejercicio,
        p.SiglaNacion as country,
        p.Nombre as salesrepresentative,
        car.ImporteEfecto,
        car.ImportePendiente,
        car.MovPosicion, p.CodigoProveedor
    FROM
        CarteraEfectos AS car
    INNER JOIN Proveedores p ON (p.CodigoProveedor =
car.CodigoClienteProveedor and p.CodigoEmpresa = car.CodigoEmpresa)
```

```
    WHERE car.Prevision = 'P'),

    v AS (SELECT ventas.*, lac.CodigoArticulo as product,
lac.CodigoFamilia as Marca , lac.Unidades FROM Ventas

    LEFT JOIN LineasAlbaranCliente lac ON (lac.CodigoDelCliente =
ventas.CodigoCliente

    and lac.CodigoEmpresa = ventas.CodigoEmpresa)),

    c AS (SELECT compras.*, lap.CodigoArticulo as product,
lap.CodigoFamilia as Marca, lap.Unidades FROM compras

    LEFT JOIN LineasAlbaranProveedor lap ON (lap.CodigodelProveedor =

    compras.CodigoProveedor and lap.CodigoEmpresa =
compras.CodigoEmpresa)),

    cv AS

    (SELECT * FROM c

    UNION

    SELECT * FROM v)

    SELECT invoiceDate, monthinvoiced,Ejercicio,

    country,salesrepresentative,product,Marca, Unidades, ImporteEfecto,

    ImportePendiente FROM cv
```

A.2. Encriptado de datos

```
def encrypt_data(value):

    if pd.isna(value):

        return value

    return hashlib.sha256(str(value).encode()).hexdigest()[:4]
```

A.3. Ventas por agente

```
def evolutionByAgent(df_filtered, mainColumnsDict):

    date_col_name = mainColumnsDict.get('date', "invoiceDate")

    agent_col_name = mainColumnsDict.get('agent', "agentecomercial")

    profit_col_name = mainColumnsDict.get('profit', "IMPLinTot")


    grouped = df_filtered.groupby([date_col_name,
agent_col_name])[profit_col_name].sum().reset_index()

    grouped_sorted = grouped.sort_values(by=[date_col_name,
agent_col_name])

    result = {

        "x_label": grouped_sorted[date_col_name].tolist(),    # Ensure
chronological order
```

```
        "y_label": grouped_sorted[profit_col_name].tolist(),
        "tag": ['Histórico' for i in range(len(grouped_sorted))],
        "group": grouped_sorted[agent_col_name].tolist()
    }
    return result
```

A.4. Mapa

```
export function mapChart(salesData, chartId = 'map-chart', tooltipText = defaultTooltipText, config = defaultMapConfig) {
  // Set up the map container and get the inner group.
  const g = setUpMap(chartId, config);
  // Also select the outer svg for event handling.
  const svg = d3.select('#' + chartId).select('svg');

  // Prepare the salesData mapping: region name (upper case) -> IMPLinTot.
  const provinciaIMPLinTot = salesData.reduce((result, d) => {
    const region = d.region.toUpperCase();
    const currentSum = result.get(region) || 0;
    result.set(region, currentSum + (+d.IMPLinTot));
    return result;
  }, new Map());

  // Load the GeoJSON data for Spain provinces.
  d3.json('/api/data/spain-provinces.geojson').then(spainProvincesData => {
    // For each province, set its IMPLinTot property based on the aggregated salesData.
    spainProvincesData.features.forEach(feature => {
      const regionName = feature.properties.name.toUpperCase();
      feature.properties.IMPLinTot = provinciaIMPLinTot.get(regionName) || 0;
    });
    // Define a color scale.
    const maxIMPLinTot = d3.max(spainProvincesData.features, d => d.properties.IMPLinTot);
    const colors = d3.schemeBlues[5];
```

```
    const thresholds = [
      maxIMPLinTot / 100,
      (maxIMPLinTot / 100) * 3,
      (maxIMPLinTot / 100) * 10,
      (maxIMPLinTot / 100) * 50,
    ];
    const colorScale = d3.scaleThreshold()
      .domain(thresholds)
      .range(colors);

    // Bind the GeoJSON features to path elements.
    const provinces = g.selectAll(".province")
      .data(spainProvincesData.features, d => d.properties.name);
```

A.5. Comparación entre marcas

```
def brandComparison(df_filtered, mainColumnsDict):


    agent_col_name = mainColumnsDict.get('agent', "agentecomercial")
    profit_col_name = mainColumnsDict.get('profit', "IMPLinTot")
    quantity_col_name = mainColumnsDict.get('quantity', "invQty")
    product_col_name = mainColumnsDict.get('product', "product")
    region_col_name = mainColumnsDict.get('region', "region")
    brand_col_name = mainColumnsDict.get('brand', "marca")
    # Group by Brand
    aggregated = df_filtered.groupby(brand_col_name).agg(
        Regiones=(region_col_name, 'nunique'),
        Agentes=(agent_col_name, 'nunique'),
        Product=(product_col_name, 'nunique'),
        Cantidad=(quantity_col_name, 'sum'),
        Ventas=(profit_col_name, 'sum')
    ).reset_index()


    selected_brands_sorted = aggregated.sort_values(by='Ventas',
ascending=False)
    result = []
    for index, row in selected_brands_sorted.iterrows():
```

```
        elements = [
            {"axis": "Propagación", "value": row['Regiones'] / aggregated['Regiones'].max(), "group": row[brand_col_name], "original_value": row['Regiones']},
            {"axis": "Agentes", "value": row['Agentes'] / aggregated['Agentes'].max(), "group": row[brand_col_name], "original_value": row['Agentes']},
            {"axis": "Productos", "value": row['Product'] / aggregated['Product'].max(), "group": row[brand_col_name], "original_value": row['Product']},
            {"axis": "Undidades Vendidas", "value": row['Cantidad'] / aggregated['Cantidad'].max(), "group": row[brand_col_name], "original_value": row['Cantidad']},
            {"axis": "Volumen Ventas", "value": row['Ventas'] / aggregated['Ventas'].max(), "group": row[brand_col_name], "original_value": row['Ventas']}
        ]
        for element in elements:
            result.append(element)

    #print(f"Total execution time: {time.time() - start_time_total:.2f} seconds")
    return result
```

A.6. Top 10 marcas

```
def get_top_brands(filtered_df, mainColumnsDict):
    year_col_name = mainColumnsDict.get('year', "yearinvoiced")
    brand_col_name = mainColumnsDict.get('brand', "marca")
    profit_col_name = mainColumnsDict.get('profit', "IMPLinTot")

    latest_year = filtered_df[year_col_name].max()
    df_latest_year = filtered_df[filtered_df[year_col_name] == latest_year]

    top_brands_df = df_latest_year.groupby(brand_col_name).agg(
        profit=(profit_col_name, 'sum')
    ).reset_index()

    top_brands_df = top_brands_df.nlargest(10, 'profit')
    top_brands_df["tag"] = "Top 10 Brands by Profit"
```

```
    result = {
        "x_label": top_brands_df[brand_col_name].tolist(),
        "y_label": [float(x) for x in top_brands_df["profit"].tolist()],
        "tag": top_brands_df["tag"].tolist(),
        "group": [int(x) for x in [latest_year] * len(top_brands_df)]
    }
    return result
```

A.7. Tendencia y predicción de ventas

```
import warnings
import pandas as pd
from statsmodels.tsa.statespace.sarimax import SARIMAX
def forecast(df, mainColumnsDict,end_year):
    date_col_name = mainColumnsDict.get('date', "invoiceDate")
    agent_col_name = mainColumnsDict.get('agent', "agentecomercial")
    profit_col_name = mainColumnsDict.get('profit', "IMPLinTot")
    # Set Date as index
    df.set_index(date_col_name, inplace=True)
    # Group data by date_col and agent_col, aggregate IMPLinTot
    grouped = df.groupby([date_col_name, agent_col_name]).agg({profit_col_name: 'sum'}).reset_index()
    # grouped.set_index(date_col_name, inplace=True)
    # Store forecasts for each agent
    forecasts = []
    with warnings.catch_warnings():
        # Ignore all ARIMA warnings
        warnings.simplefilter("ignore")
        # Iterate over each agent and fit a model
        for agent in grouped[agent_col_name].unique():
            agent_data = grouped[grouped[agent_col_name] == agent]
            agent_data.set_index(date_col_name, inplace=True)
            agent_data = agent_data.resample('M').sum()
            model = SARIMAX(agent_data[profit_col_name], order=(1, 1, 1), seasonal_order=(1, 1, 1, 12))
            sarima_model = model.fit(disp=False)
```

```
            forecast = sarima_model.get_forecast(steps=12)
            forecast_df = forecast.conf_int(alpha=0.05)
            forecast_df[profit_col_name] = forecast.predicted_mean
            forecast_df[agent_col_name] = agent
            forecast_df.reset_index(inplace=True)
            forecasts.append(forecast_df[['index',     profit_col_name,
agent_col_name]])
    # Concatenate all forecasts
    forecast_df = pd.concat(forecasts)
    forecast_df.rename(columns={'index': date_col_name}, inplace=True)
    # Combine both data
    combined = pd.concat([grouped, forecast_df])
    # Compute overall mean forecast
    mean_forecast                                                    =
forecast_df.groupby(date_col_name)[profit_col_name].mean().reset_index
()
    mean_forecast[agent_col_name] = 'Media Global'
    # Combine both data
    combined_all = pd.concat([combined, mean_forecast])
    combined_sorted = combined_all.sort_values(by=date_col_name)
    combined_sorted.loc[combined_sorted[date_col_name].dt.year
>end_year, agent_col_name] = 'Prediccion'
    return combined_sorted
def filterForecast(df, mainColumnsDict, filters):
    date_col_name = mainColumnsDict.get('date', "invoiceDate")
    agent_col_name = mainColumnsDict.get('agent', "agentecomercial")
    # Obtener valores de filtro
    start_year, end_year = 2020, 2024
    years = filters.get('years', None)
    print(filters)
    end_year +=1
    selected_agents = filters.get('agents', [])   # Lista de agentes
seleccionados
    print(selected_agents)
    if years:
        start_year, end_year = [int(year) for year in years.split('-')]
    # Asegurar que selected_agents es una lista válida
```

```
    if isinstance(selected_agents, str):
        selected_agents = [selected_agents]
    # Agregar 'Media Global' a los agentes seleccionados si no está
    if 'Media Global' not in selected_agents:
        selected_agents.append('Media Global')
    # Filtrar por años y agentes
    df_filtered = df[
        (df[date_col_name].dt.year >= start_year) &
        (df[date_col_name].dt.year <= end_year) &
        (df[agent_col_name].isin(selected_agents))
    ]
    print(df_filtered)
    return df_filtered
def trend(df_filtered, mainColumnsDict):
    date_col_name = mainColumnsDict.get('date', "invoiceDate")
    agent_col_name = mainColumnsDict.get('agent', "agentecomercial")
    profit_col_name = mainColumnsDict.get('profit', "IMPLinTot")
    result = {
        "x_label":          df_filtered[date_col_name].dt.strftime('%Y-
%m').tolist(),
        "y_label": df_filtered[profit_col_name].tolist(),
        "tag":  ['Histórico'  if  x  <=  pd.Timestamp('today')  else
'Pronóstico' for x in df_filtered[date_col_name]],
        "group": df_filtered[agent_col_name].tolist()
    }
    return result
```

A.8. Filtros

```
def mainFilter(df, mainColumnsDict ,filters=None):
    agent_col_name = mainColumnsDict.get('agent', "agentecomercial")
    year_col_name = mainColumnsDict.get('year', "yearinvoiced")
    date_col_name = mainColumnsDict.get('date', "invoiceDate")
    brand_col_name = mainColumnsDict.get('brand', "marca")
```

```
    if not filters:
        return df
    filtered_df = df.copy()

    # Time
    years = filters.get('years', None)
    if years:
        start_year, end_year = [int(year) for year in years.split('-')]
        if  year_col_name in list(filtered_df.columns):
            filtered_df  =  filtered_df[(filtered_df[year_col_name]  >= start_year) & (filtered_df[year_col_name] <= end_year)]
        else:
            filtered_df                                                 = filtered_df[(filtered_df[date_col_name].dt.year   >=   start_year)   & (filtered_df[date_col_name].dt.year <= end_year)]
    # Agents
    lista_agentes = filters.get('agents', None)
    if lista_agentes:
        filtered_df                                                 = filtered_df[filtered_df[agent_col_name].isin(lista_agentes)]
    # Brands
    brand_list = filters.get('brands', None)
    if brand_list:
        filtered_df                                                 = filtered_df[filtered_df[brand_col_name].isin(brand_list)]

    return filtered_df
def brandComparisonFilter(df, mainColumnsDict ,filters=None):
    agent_col_name = mainColumnsDict.get('agent', "agentecomercial")
    year_col_name = mainColumnsDict.get('year', "yearinvoiced")
    date_col_name = mainColumnsDict.get('date', "invoiceDate")
    brand_col_name = mainColumnsDict.get('brand', "marca")

    # Time
    filtered_df = df.copy()
    years = filters.get('years', None)
```

```
    if years:
        start_year, end_year = [int(year) for year in years.split('-')]
        if  year_col_name in list(filtered_df.columns):
            filtered_df  =  filtered_df[(filtered_df[year_col_name]  >=
start_year) & (filtered_df[year_col_name] <= end_year)]
        else:
            filtered_df                                                 =
filtered_df[(filtered_df[date_col_name].dt.year   >=   start_year)   &
(filtered_df[date_col_name].dt.year <= end_year)]
    selected_brands = filters.get('brands', [])
    if selected_brands == []:
        selected_brands = filtered_df[brand_col_name][0:3]


    # Asegurar que selected_agents es una lista válida
    if isinstance(selected_brands, str):
        selected_brands = [selected_brands]


    # Filtrar por años y agentes
    df_filtered                                                         =
filtered_df[filtered_df[brand_col_name].isin(selected_brands)]
    return df_filtered
```