

TRABAJO FINAL DE MÁSTER

IA EN LA GESTIÓN DE CITAS MÉDICAS:
INNOVANDO EN OPTIMIZACIÓN DE AGENDAS
MÉDICAS Y ATENCIÓN AL CLIENTE.

Descripción breve

La utilización de Inteligencia Artificial (IA) en los Centros Médicos para optimizar la asignación de citas médicas, con el objetivo de maximizar los servicios de salud prestados y la rentabilidad de los recursos disponibles, así como para mejorar la atención al cliente con un Asistente Virtual que aumenta la capacidad de gestión del Centro Médico y disminuye los tiempos de respuesta.

Grupo 3

Hernando Acevedo Aguilar
Michelle Alexandra Chicaiza Anrrango
Luis Marcelo Ortiz Carinao
Sergio Valdueza Lozano

1 octubre 2024

Resumen ejecutivo del proyecto

Uno de los principales problemas de los servicios de salud tanto en España como en Latinoamérica son los tiempos de espera para consultas médicas de especialidades y el incumplimiento de estas citas, debido a múltiples factores, tales como la brecha de especialistas (1), y la inasistencia a consultas médicas, que puede llegar al 20% del total de usuarios (2) (3).

Para abordar esta problemática, proponemos integrar herramientas de inteligencia artificial (IA) con el fin de mejorar la asignación de citas médicas. Esto se lograría mediante la predicción de la probabilidad de inasistencia ("No Show") utilizando técnicas de Machine Learning (ML) y Deep Learning (DL). Al calcular dicha probabilidad y crear agendas médicas más compactas, se podría optimizar el uso de los recursos de los centros médicos, reduciendo tanto la inasistencia de pacientes como las listas de espera. El proyecto "Aplicación de Inteligencia Artificial en la Gestión de Citas Médicas: Optimización y Reducción de Ausencias" aborda este reto mediante el desarrollo de un sistema inteligente que optimiza la asignación de citas y minimiza tiempos de inactividad de médicos. El proyecto se estructura en cuatro fases interconectadas: Predicción de asistencia a citas médicas, Implementación de un sistema de overbooking inteligente, Creación de un asistente virtual para la gestión de citas, e Integración para mejora continua del sistema. Los resultados obtenidos son notablemente prometedores.

En la primera fase, se desarrollan modelos predictivos de Deep Learning (NeuralNet y TabNet) con un AUC significativamente superior a los métodos tradicionales (modelos de Machine Learning), alcanzando un AUC (Área Bajo la Curva ROC) de 0.74.

Con la implementación del sistema de overbooking en la segunda fase se alcanza el objetivo propuesto de disminuir listas de espera, y se evidencia un potencial de ahorro económico, mostrando que un centro médico con 20 consultas médicas podría ahorrar casi 2 millones de euros anuales si no tiene implantado un sistema de overbooking en la programación de citas, o unos 650mil euros anuales si tuviese el mejor de los sistemas tradicionales que permiten cierto grado de overbooking.

La tercera fase introduce a "Basilio", asistente virtual de vanguardia de tercera generación que permite una gestión eficaz de citas y la interacción con pacientes. Al compararlo con la gestión de citas con personal humano, reduce los costes operativos y permite una respuesta inmediata a usuarios.

En conjunto, los resultados del proyecto son positivos, avanzando a objetivos de optimización y eficiencia y abriendo nuevas posibilidades para aplicación de inteligencia artificial en el sector sanitario. La reducción de tiempos de espera, la optimización de recursos y la mejora en la accesibilidad a servicios médicos son posibles avances que podrían evidenciarse en una aplicación potencial del proyecto.

Sin embargo, también reconoce los desafíos inherentes a la implementación de tecnologías avanzadas en el sector sanitario, abordando con rigor aspectos como la precisión de las predicciones, la ética en el manejo de datos médicos y la necesidad de validación en entornos reales.

Contenido

Resumen ejecutivo del proyecto	1
Introducción	3
Núcleo del proyecto	5
Definición del proyecto y análisis de viabilidad	5
Descripción detallada de la empresa	5
Análisis interno y externo	6
Explicación detallada del proyecto	7
Fijación de los objetivos generales y específicos del proyecto	11
Planificación	12
Desarrollo del proyecto	12
Fase 1. Predicción de Asistencia a Citas Médicas	12
Fase 2. Implementación de un sistema de overbooking	37
Fase 3. Creación de un Asistente Virtual basado en el Procesamiento de Lenguaje Natural (NLP) ..	51
Fase 4. Proceso de Integración del Sistema.	62
Conclusiones	64
Bibliografía	67
Tabla de figuras.....	69
Anexos	71
Anexo 1 – Cronograma del proyecto	71
Anexo 2 – Estimación de recursos económicos	74
Anexo 3 – Prompt asistente	75

Introducción

MediAgenda Solutions, S.L. (empresa ficticia para fines académicos), es una empresa líder del sector de la salud, especializada en uso intensivo de Inteligencia Artificial para administración y gestión de agendas médicas para hospitales y centros médicos en España y Latinoamérica. Nuestro proyecto se centra en gestionar agendas de visita a especialistas, con la misión de optimizar la asignación de horarios médicos, reducir los tiempos de espera y mejorar la calidad de la atención médica.

En España, para el acceso a una atención especializada, los pacientes esperan una media de hasta 79 días para ser atendidos, rango entre 22 y 107 días (4). En Latinoamérica particularmente en Chile, a 31 de diciembre de 2022 se tuvo 61.191 garantías retrasadas, con promedio de 156.5 días de retraso por garantía (5). Estos retrasos se explican por múltiples factores, como la brecha de especialistas (1) y la inasistencia que en algunos centros alcanza hasta el 20% (2) (3). La falta de asistencia a citas, o "No Show", es un obstáculo que entorpece la eficiencia operativa, genera desequilibrios de programación de los médicos y prolonga las listas de espera para ser atendido por un especialista.

Para abordar esta problemática, se propone integrar herramientas de inteligencia artificial para mejorar la asignación de citas médicas mediante la predicción de la probabilidad de inasistencia usando técnicas de Machine Learning y Deep Learning. Con base en esta probabilidad, se optimizan las agendas médicas mediante sistema de overbooking, similar al utilizado en otros sectores (aviación, hoteles, etc.), para mejorar la ocupación de los recursos, y reducir los retrasos en las consultas. Además, se desarrolla una interfaz de usuario que facilita la concertación y confirmación de citas de forma eficiente y amigable. El Proyecto se desarrolla en 4 fases:

Fase 1: Predicción de asistencia a citas médicas.

La primera fase, el núcleo fundamental del mismo consta de varias subfases. Inicialmente, se desarrolla un modelo de IA para predecir la asistencia a citas médicas se recopilan datos y se hace un análisis exploratorio para identificar variables clave y crear características significativas. Se aplica particionamiento de datos siguiendo distintos criterios para entrenar y evaluar los modelos, se usa regresión logística, árbol de decisión y redes neuronales, realizando optimización de los modelos con búsqueda exhaustiva de los mejores parámetros. Los modelos se clasifican según su rendimiento, seleccionando los más efectivos en cada particionamiento para implementarlos en la gestión de overbooking. El proceso se realiza con el uso de Python lenguaje de programación principal, aprovechando entornos de desarrollo como Google Colab, Visual Studio Code y Jupyter Notebook.

Se desarrollan y evalúan varios modelos de machine learning, destacando TabNet como el más efectivo. Este modelo alcanzó un AUC de 0.74, demostrando una capacidad discriminativa superior a los métodos tradicionales. TabNet también logró el mejor F1-score, llegando a 0.49, siendo un resultado más óptimo entre precisión y recall. La implementación de técnicas de feature engineering y balanceo de datos como SMOTE-ENN (Synthetic Minority Over-sampling Technique combinado con Edited Nearest Neighbors) y ADASYN (Adaptive Synthetic Sampling) contribuyó significativamente a estos resultados. Destaca notablemente el valor de recall obtenido (78,95%) utilizando una red neuronal en el conjunto de datos con el dataset completo, sin eliminación de variables y aplicando ADASYN, lo cual es particularmente valioso para minimizar los falsos negativos en la predicción de no-shows.

Fase 2: Implementación de un sistema de overbooking.

En la segunda fase, se utilizan las predicciones de asistencia para optimizar las agendas médicas, posibilitando la asignación de citas múltiples a un mismo slot, generando cupos de overbooking, para maximizar el uso de recursos sin generar tiempos de espera excesivos ni sobrecargar los servicios médicos. Se definen los costes de una agenda en función de los tiempos de inactividad en la clínica por falta de pacientes a atender, los tiempos de espera de los pacientes y los tiempos de trabajo extra para atender a los pacientes rezagados. Dichos costes se evalúan primeramente en las agendas médicas obtenidas con las reglas de asignación de citas tradicionales, y a posteriori con agendas médicas obtenidas mediante una función que tiene en cuenta probabilidades de asistencia de los pacientes, obtenidas con el modelo de predicción de la Fase 1.

La implementación del sistema de overbooking basado en las predicciones de la Fase 1 demuestra una mejora significativa en la eficiencia y reducción de costos. El escenario más efectivo logra una reducción del 25% en el costo total del sistema en comparación con el escenario tradicional sin overbooking. Adicional, en términos económicos, se estima un ahorro potencial de hasta 2 millones de euros anuales para un centro médico con 20 consultas que no esté aplicando hasta el momento ningún sistema de overbooking en la programación de citas, o de 650 mil euros si estuviese aplicando el mejor de los sistemas tradicionales (los cuales sí permiten cierto grado de overbooking).

Fase 3: Creación de un asistente virtual basado en NLP

En la tercera fase, se desarrolla un Asistente Virtual de tercera generación basado en un Large Lengual Model (LLM) para gestionar citas y proporcionar información sobre especialidades médicas y medicamentos, además de procesar imágenes. El sistema se integra vía API (*Application Programming Interface*) con Google Sheets para la gestión de datos y con la API de Telegram para la interfaz de usuario. El asistente facilita la accesibilidad del paciente, y también optimiza los recursos del centro médico, disminuyendo costes operativos y, contribuyendo a la modernización de los servicios de salud.

Se desarrolla con éxito "Basilio", un asistente virtual de tercera generación utilizando el modelo GPT-4o y técnicas de Retrieval Augmented Generation (RAG). Basilio demuestra capacidades en la gestión de citas médicas, incluyendo programación, modificación y cancelación de citas. El asistente logró procesar y responder a consultas de usuarios, manteniendo precisión en la información proporcionada sobre especialidades médicas y medicamentos. La integración con Telegram permite una mejor accesibilidad.

Fase 4: Consideraciones generales para la Integración del Sistema

En la cuarta fase se describe, a nivel teórico, un refinamiento del Sistema, el cual integraría la base de datos que se utiliza en el Asistente Virtual con modelos predictivos de asistencia y de optimización para asignación de citas, cerrando el círculo con retroalimentación periódica a los modelos de IA, que les ayude a mejorar la precisión en las predicciones y, por ende, en la optimización de agenda médica obtenida.

Se definen las bases de un proceso de refinamiento continuo que integra la base de datos de la interfaz creada para gestionar las citas con los modelos predictivo y de optimización en la programación. Este proceso permite la actualización periódica de los modelos de IA, incluyendo la posibilidad de introducir nuevas características predictoras relevantes, mejorando la precisión de las predicciones de asistencia y, por ende, la optimización en la preparación de agendas médicas de menor coste y más atención médica. Además, la integración del sistema de gestión de citas con la interfaz de usuario facilita una experiencia fluida y eficiente para los pacientes, otorgando funcionalidades paralelas que mejoran la experiencia de los usuarios y ayudan también a disminuir la carga de trabajo en las consultas médicas.

Resultados clave:

- **Reducción de Tiempos de Espera Global:** La implementación del sistema de overbooking, junto con las predicciones precisas del modelo de IA, permitiría disminuir los tiempos de espera para acceder a una consulta médica.
- **Optimización de Recursos Médicos:** La utilización eficiente de los recursos médicos a través del sistema de overbooking y la mejora en la programación de citas contribuye a un uso más efectivo y racional de los mismos, generando ahorros económicos y operativos muy importantes.
- **Mejora Atención al Cliente:** La puesta en marcha de un Asistente Virtual de 3^a generación permite interactuar con el Centro médico con un nuevo canal de comunicación, permitiendo una gestión ágil y práctica de las citas médicas, consultas de prescripciones médicas, asesoramiento inicial para la concertación de visitas médicas.

En definitiva, los resultados obtenidos pueden llevar a una mejora significativa en la eficiencia operativa de los centros médicos, una reducción en los costos y una mayor satisfacción del paciente.

El proyecto resalta el potencial de la inteligencia artificial para mejorar la atención médica, ya que, con la colaboración adecuada entre profesionales de la salud y expertos en tecnología, se pueden lograr avances en la eficiencia operativa y la experiencia de los pacientes.

Núcleo del proyecto

Definición del proyecto y análisis de viabilidad

Descripción detallada de la empresa

Se plantea la creación de MediAgenda Solutions, S.L. como empresa líder en el sector de la salud, especializada en la administración inteligente de agendas médicas para hospitales y centros médicos en España y Latinoamérica.

Nos enfrentamos al desafío crítico de los largos tiempos de espera en la atención médica, una problemática que impacta tanto a los pacientes, quienes experimentan retrasos significativos en su atención, como a los profesionales de la salud, que deben hacer frente a agendas sobrecargadas y recursos limitados.

Un ejemplo ilustrativo de esta problemática se encuentra en el barómetro sanitario realizado en España por el Ministerio de Sanidad, Consumo y Bienestar Social (6) en colaboración con el Centro de Investigaciones Sociológicas. Según los datos recogidos en el informe correspondiente al año 2023, el 27.2% de los ciudadanos reportaron haber esperado "11 días o más" desde que solicitaron la cita hasta que fueron atendidos por el médico de familia, evidenciando así los prolongados tiempos de espera que afectan a la población.

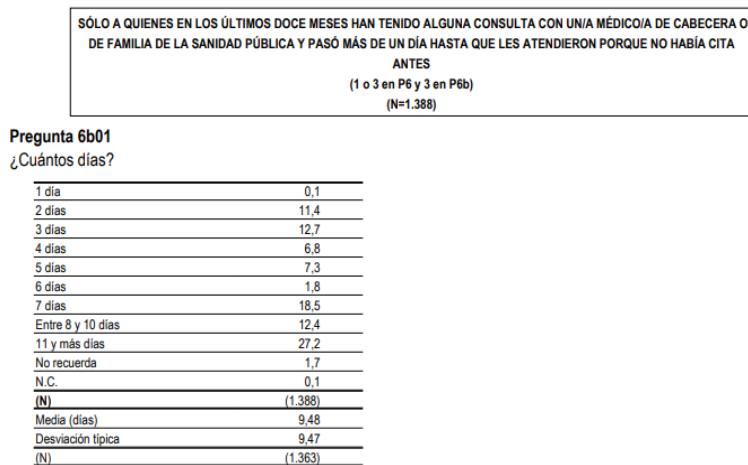


Figura 1. Tiempo de espera en días para consulta a médico de familia.

Con el objetivo de abordar este problema, hemos centrado nuestros esfuerzos en el desarrollo de soluciones innovadoras que optimicen la gestión de citas médicas mediante la aplicación de tecnologías avanzadas, especialmente inteligencia artificial. Nuestros avanzados algoritmos de machine learning analizan una amplia gama de datos, desde historiales médicos hasta patrones de comportamiento previos, para predecir con precisión la probabilidad de inasistencia de los pacientes a sus citas médicas. Esta información se utiliza para ajustar de manera inteligente las agendas médicas, maximizando así la eficiencia de los recursos médicos disponibles y reduciendo los tiempos de espera para los pacientes.

Además de los desafíos asociados con los largos tiempos de espera en la atención médica, otro problema significativo que enfrentan tanto pacientes como profesionales de la salud es el alto porcentaje de absentismo en las citas médicas. Este fenómeno no solo genera costos económicos para los sistemas de salud, sino que también puede afectar negativamente la calidad y eficiencia de la atención médica.

Conscientes de esta problemática, MediAgenda Solutions, S.L. desarrolla una solución innovadora que aborda directamente el problema del absentismo. Nuestra plataforma utiliza algoritmos avanzados

que permiten una mejor planificación de los recursos médicos con una oportunidad de atención más eficiente.

La eficacia de las innovaciones tecnológicas en el ámbito médico se ha demostrado en términos de ahorro de tiempo tanto para los pacientes como para los profesionales de la salud, lo que se traduce en una reducción de costos. Estas innovaciones no solo mejoran la organización de los horarios y una utilización óptima de los recursos hospitalarios, sino que también permiten una actualización inmediata de la información del paciente y una mayor flexibilidad en la programación de citas.

La empresa tiene su sede principal en Barcelona, España, desde donde coordina sus operaciones y desarrollo tecnológico. Sin embargo, su alcance abarca a nivel nacional e internacional, con el objetivo de ofrecer soluciones a los problemas de tiempos de espera en los servicios de salud en países de habla hispana, especialmente en España y Latinoamérica.

Análisis interno y externo

Con el propósito de obtener una visión detallada y completa de la posición y perspectivas de MediAgenda Solutions, S.L. en el sector de la salud, se realiza un análisis tanto de los factores internos como externos que influyen en la empresa.

Análisis interno

Innovación Tecnológica: MediAgenda Solutions, S.L. se destaca por fomentar una cultura de innovación y colaboración, y se mantiene actualizada en las últimas tendencias y desarrollos en el campo de la inteligencia artificial y la tecnología de la salud.

Equipo Multidisciplinario: La empresa cuenta con un equipo altamente calificado y multidisciplinario, conformado por profesionales en áreas como inteligencia artificial, medicina, gestión de proyectos, diseño de experiencia de usuario (UX/UI) y desarrollo de software con el objetivo de desarrollar y ofrecer una solución innovadora y eficaz para abordar el problema de los largos tiempos de espera en las consultas médicas especializadas. Esta diversidad de talentos permite una visión integral en el desarrollo de sus productos y servicios.

Alianzas Estratégicas: MediAgenda Solutions, S.L. ha establecido alianzas estratégicas con instituciones médicas, centros de salud y organizaciones del sector para colaborar en la implementación y mejora continua de sus soluciones. Estas alianzas fortalecen su posición en el mercado y les permiten acceder a una base de clientes potenciales más amplia.

Análisis externo

Demandas en Crecimiento: La creciente demanda de soluciones para reducir los tiempos de espera en los servicios de salud representa una oportunidad clave para MediAgenda Solutions, S.L. La necesidad de optimizar la gestión de citas médicas es un problema extendido en el sector, tanto en España como en Latinoamérica, lo que brinda un mercado potencialmente amplio para sus servicios.

Competencia: Aunque MediAgenda Solutions, S.L. es líder en la integración de inteligencia artificial en la gestión de citas médicas, enfrenta competencia de otras empresas que ofrecen soluciones similares o alternativas tradicionales. La capacidad de innovación y diferenciación será crucial para mantener su posición en el mercado.

Regulaciones y Normativas: La empresa opera en un sector altamente regulado, sujeto a normativas específicas en materia de protección de datos, seguridad y calidad de servicios de salud. Cumplir con estas regulaciones es fundamental para ganar la confianza de clientes y usuarios finales, así como para mantener la reputación y credibilidad de la empresa.

Explicación detallada del proyecto

Para la ejecución de este proyecto se plantean 4 fases o retos que nos ayuden a conseguir el objetivo planteado:

Fase 1: Realizar un modelo de IA capaz de predecir la Asistencia o No Asistencia de los pacientes a las citas médicas.

Esta fase es esencial ya que sienta las bases para el desarrollo de soluciones efectivas en las etapas posteriores del proyecto.

- **Elección, Recolección y Preparación de datos.** Se realizó una búsqueda de una base de datos que contiene información sobre las citas médicas programadas en un servicio de salud, así como datos relacionados con los usuarios que las solicitan. Se elige una base de datos que incluye una etiqueta que indica si el paciente asistió o no a su cita médica (Show – No Show), ya que esta información es esencial para el entrenamiento y la evaluación del modelo de inteligencia artificial.
- **Análisis Exploratorio de los Datos (EDA).** Se realizó un exhaustivo EDA para comprender la estructura, la distribución y las relaciones dentro del conjunto de datos. Durante este proceso, se identificaron y prepararon las variables predictoras relevantes para la predicción de inasistencia a las citas médicas. Esto implica la limpieza de datos para tratar inconsistencias o valores faltantes, así como explorar las variables presentes en la base de datos en busca de relaciones, tendencias o anomalías. Este paso es crucial para garantizar la calidad y la confiabilidad de los datos utilizados en el entrenamiento del modelo.
- **Feature Engineering.** El proceso de Feature Engineering desempeña un papel fundamental en la preparación de los datos para la construcción de modelos de inteligencia artificial. Durante esta etapa, transformamos los datos brutos del dataset en características significativas que permiten a los modelos aprender patrones y realizar predicciones precisas. Este proceso se divide en varias etapas clave que incluyen la extracción de características, la transformación de tipos de datos, la creación de nuevas características, la selección de características relevantes, y la normalización y manejo de los datos faltantes.

Durante la **extracción de características**, se identificaron datos relevantes del dataset original, como la información demográfica de los pacientes y datos relacionados con el historial médico y las visitas anteriores, los cuales son importantes para predecir la asistencia o no asistencia a las citas médicas.

La **transformación de tipos** de datos garantiza que los datos estén en un formato adecuado para su procesamiento por parte de los modelos de inteligencia artificial. Además, se crearon nuevas características que pueden incluir información climática, ubicación geográfica, y variables relacionadas con el historial del paciente, entre otras.

La **selección de características** se basa en la evaluación de su correlación con la variable objetivo (No Show), asegurando que solo se utilicen aquellas que tengan un mayor impacto en la predicción.

Finalmente, se aplican **técnicas de normalización, reducción de dimensionalidad**, manejo de **variables categóricas** y **balanceo de clases** para garantizar la calidad y eficacia del modelo de inteligencia artificial.

Este proceso de Feature Engineering sienta las bases para la construcción de modelos predictivos precisos en la gestión de citas médicas, permitiendo una mejor comprensión de los factores que influyen en la asistencia o no asistencia a las citas médicas.

- **Entrenar diferentes modelos de machine learning.** Se entrenaron diferentes modelos de machine learning, aplicando diversas técnicas y estrategias para asegurar una predicción precisa de la asistencia de los pacientes a sus citas médicas. Para el entrenamiento de los modelos de machine learning, se seguirá un enfoque metodológico detallado:
 - **Selección del conjunto de prueba.** Dado que no hay un conjunto de datos de prueba separado, se reservará una parte del conjunto original. Este subconjunto mantendrá la distribución de clases de original. La elección de este conjunto no se hará de forma aleatoria, se ordenará el dataset por fecha de cita programada, en orden descendente, seleccionando sólo la última cita de cada paciente, de forma que no se repita paciente alguno.
 - **Hipótesis y segmentación de datos.** Durante la preparación de los datos, se considerarán varias hipótesis para crear conjuntos de datos específicos que mejoren la capacidad predictiva de los modelos:
 - Conjunto de datos completo.
 - Pacientes sin condiciones médicas.
 - Pacientes de edad entre los 5 y 30 años.
 - Pacientes con citas programadas para otro día.
 - Pacientes de barrios con centro médico.
 - Pacientes segmentados por grupos de edad:
 - Niños: Menores de 12 años
 - Adolescentes: 13 - 18 años
 - Jóvenes adultos: 19 - 35 años
 - Adultos: 36 - 64 años
 - Adultos mayores: 65 años en adelante
 - **Entrenamiento de modelos de machine learning.** Para cada uno de los conjuntos de datos generados, se entrenaron varios modelos de machine learning. Entre los modelos seleccionados se incluyen regresión logística, árboles de decisión y redes neuronales. Durante el entrenamiento, se realizaron búsquedas de hiperparámetros y validaciones cruzadas para optimizar el rendimiento de cada modelo.
 - **Evaluación y optimización.** Una vez finalizado el entrenamiento, se procedió a una evaluación de los datos de entrenamiento y prueba. Se calcularon diversas métricas de rendimiento para cada modelo y dataset, incluyendo Accuracy, Precisión, Recall, F1-Score y Curva ROC.

Fase 2: Implementación de un Sistema de Overbooking en la Programación de Citas

El objetivo de esta 2^a fase es maximizar el uso de los recursos de los centros médicos (minimizar los tiempos de inactividad) buscando un balance en los costos asociados con la implementación del sistema de overbooking, es decir, considerando también los tiempos de espera de los pacientes y las horas extra realizadas por el personal sanitario.

- **Estudio de Bibliografía Relacionada.** Se ha revisado extensivamente la bibliografía sobre los costos provocados por la infrautilización o sobreutilización de recursos médicos, así como aquellos relacionados con la reducción en la calidad del servicio debido a los tiempos de espera de los pacientes. Se identificaron diversas formas de medir los costos, variables que impactan en ellos y métodos de asignación de citas para reducirlos.
- **Elección de Hipótesis y Fórmula para el Cálculo del Coste del Sistema.** Se seleccionaron hipótesis que acotaron los posibles escenarios a desarrollar, ajustándolos al conjunto de datos elegido en la Fase 1 para el Modelo Predictivo de Asistencia. Estas hipótesis definieron y delimitaron diferentes casos de estudio y generaron restricciones para el modelo de optimización de

overbooking. Paralelamente, se eligió la fórmula para el cálculo del Coste del Sistema, función a minimizar para optimizar la asignación de citas médicas.

- **Cálculo del Coste del Sistema con Método Tradicional.** Se realizó el cálculo aritmético puro de los Costes del Sistema utilizando diferentes reglas tradicionales de asignación de citas, sin sistema de optimización alguno, y con diferentes grados de overbooking. Cada caso se ejecutó con el mismo conjunto de pruebas de la Fase 1 para garantizar la robustez y comparabilidad entre los diferentes escenarios.
- **Cálculo del Coste del Sistema con Función Basada en Predicciones de Asistencia.** Se realizó el mismo cálculo aritmético que en el punto anterior, pero esta vez realizando la asignación de citas mediante una función que tuviese en cuenta la probabilidad de asistencia de los pacientes ya programados para una fecha determinada. Todos los casos calculados, diferenciándose entre sí por los argumentos pasados a la función mencionada, fueron aplicados sobre el mismo conjunto de pruebas usado en el Método Tradicional, por lo que se garantizó la comparabilidad de resultados, y que las predicciones usadas para asignar las citas no fueran aprendidas previamente por el Modelo de Predicción.
- **Comparativa de Costes y Conclusiones en la optimización del overbooking.** Se compararon los Costes obtenidos en todos los escenarios comentados, detallando las conclusiones.

Fase 3: Creación de un Asistente Virtual basado en el Procesamiento de Lenguaje Natural (NLP).

En esta fase del proyecto, se desarrolla **Basilio**, un asistente virtual avanzado diseñado para optimizar la gestión de citas médicas en centros de salud. Basilio emplea técnicas de Procesamiento de Lenguaje Natural (NLP) y se basa en un Large Language Model, específicamente un asistente de la API de Open AI basado en **GPT-4o**, para ofrecer una experiencia interactiva eficiente y amigable a través de plataformas de mensajería como Telegram.

- **Objetivos generales**
 - **Gestión de citas médicas:** Facilitar la solicitud, modificación y cancelación de citas médicas de manera rápida y eficiente, reduciendo la carga administrativa y mejorando la disponibilidad de recursos médicos.
 - **Proporcionar información relevante:** Ofrecer detalles sobre disponibilidad de citas, servicios médicos, especialidades y resolver preguntas frecuentes relacionadas con el proceso de citas médicas.
- **Enfoque del Asistente Virtual**
 - Basilio se integra con técnicas avanzadas de **NLP** y **RAG** para proporcionar respuestas contextualmente relevantes y precisas. Este enfoque permite manejar interacciones complejas y ofrece un servicio accesible y eficiente a los pacientes.
- **Beneficios clave**
 - **Reducción de errores humanos:** Automatización del proceso de asignación de citas, minimizando errores manuales.
 - **Disponibilidad 24/7:** Los pacientes pueden gestionar sus citas en cualquier momento, sin restricciones de horario.
 - **Optimización de recursos:** Mejor utilización de los recursos médicos.
 - **Mejora de la experiencia del paciente:** Interacción más fluida y amigable, aumentando la satisfacción del paciente con el servicio.
- **Descripción general del funcionamiento:** Basilio permite a los pacientes:

- **Agendar citas:** Solicitar nuevas citas especificando especialidad, fecha y hora deseadas.
 - **Consultar citas:** Verificar las citas programadas proporcionando información de identificación.
 - **Modificar o cancelar citas:** Cambiar la fecha y hora de citas existentes o cancelarlas.
 - **Orientación médica:** Ofrecer información sobre qué especialidad consultar según los síntomas descritos.
- **Integraciones tecnológicas**
 - **API de OpenAI:** Tras revisar diversas plataformas de chatbots como Dialogflow, Amazon Lex, Microsoft Bot Framework y developers OpenAI, se decide utilizar la API de OpenAI debido a su avanzada capacidad para generar respuestas naturales y precisas, y su flexibilidad para la integración de su asistente. Se implementa el modelo GPT-4o para facilitar interacciones avanzadas, generando respuestas contextuales a través de instrucciones de sistema que establecen el contexto y comportamiento general del asistente. También se aplicaron técnicas de file search para acceder a la información relevante en la base de datos de citas médicas.
 - **Google Sheets:** Utilizado para almacenar y gestionar datos relacionados con citas médicas y disponibilidad de médicos. La integración con esta base de datos mediante API permite a Basilio acceder y actualizar la información sobre disponibilidad de citas, horarios de médicos y especialidades, asegurando respuestas precisas.
 - **Telegram:** Plataforma de mensajería utilizada para la interacción con los pacientes, proporcionando un acceso conveniente y en tiempo real.
 - **Procesamiento de imágenes:** Integración de librerías para identificar medicamentos a partir de imágenes enviadas por los usuarios.

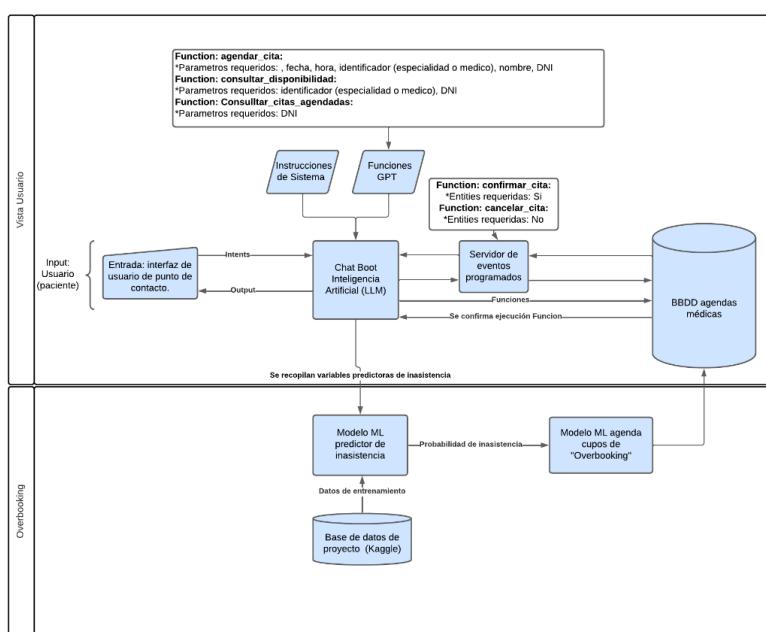


Figura 2. Diagrama de flujo de la propuesta de asistente virtual

Basilio, como asistente virtual de tercera generación, representa un avance significativo en la gestión de citas médicas. Su capacidad para interactuar de manera eficiente y amigable a través de una interfaz "text to text", junto con sus técnicas avanzadas de NLP y RAG, optimiza tanto la experiencia del paciente como la eficiencia operativa del hospital.

Fase 4: Integración de todo el Sistema.

Se describió, de forma conceptual, un proceso de refinamiento del sistema que integrase la base de datos de la interfaz creada para gestionar las citas de los pacientes (Fase 3) con los modelos predictivo y de optimización generados en las Fases 1 y 2. Algunos de los temas tratados para la integración mencionada fueron los siguientes:

- Selección de posibles nuevas características relevantes provenientes de los datos recopilados con el Asistente Virtual.
- Adhesión de estas nuevas características predictivas al dataset de entrenamiento elaborado en la Fase 1.
- Mejora de los modelos predictivos de la fase 1 y de optimización de la Fase 2 con nueva información disponible en un entorno de producción.
- Fusión del análisis de probabilidad de inasistencia y de la programación de citas con una base de datos integrada a una interfaz de usuario.

Fijación de los objetivos generales y específicos del proyecto

El proyecto tiene como **objetivo general mejorar la eficiencia y reducir los costos asociados con la gestión de citas médicas** mediante el uso de técnicas de Inteligencia Artificial.

Para alcanzar el objetivo general mencionado anteriormente, se plantean los siguientes **objetivos generales y específicos**:

1. **Desarrollar un exhaustivo EDA** del dataset escogido para el entrenamiento del modelo, de forma que se consiga:
 1. Limpiar el dataset de datos, vacíos, erróneos y aislados (Obj1.1).
 2. Fusionar o reformular variables que recojan información más relevante con respecto a la etiqueta (Obj1.2).
 3. Definir nuevas variables, ampliando el dataset original, que aporten información adicional relevante para la predicción (Obj1.3).
2. **Obtener el mejor modelo predictivo de No Asistencia a una cita médica**, entrenando, evaluando y comparando diferentes algoritmos de aprendizaje automático, de forma que se consiga:
 1. Optimizar la búsqueda de los mejores parámetros para cada algoritmo seleccionado, mediante el uso de GridSearch (Obj2.1).
 2. Obtener la mejor AUC para el modelo de predicción, buscando el mejor balance entre precisión y recall. Esto implica minimizar los falsos positivos (predicciones incorrectas de que un paciente no asistirá a una cita cuando sí lo hace) y los falsos negativos (predicciones incorrectas de que un paciente asistirá a una cita cuando no lo hace) de la forma más balanceada posible (Obj2.2).
3. **Estudiar cómo se generan las Agendas Médicas y que costes pueden tener asociados cuando se presentan deficiencias en la asignación de citas**, de forma que se logre:
 1. Encontrar y optimizar una Regla de Asignación de pacientes a las citas médicas que tenga en cuenta las predicciones de asistencia de estos (Obj3.1).
 2. Obtener una fórmula de costes asociados a una atención médica ineficiente, identificando las variables que afectan a dichos costes, y definiendo las hipótesis que mejor se ajustan al problema planteado (Obj3.2).

3. Minimizar los costes producidos por agendas de atención médica inefficientes, poniendo especial relevancia en la maximización del uso de los recursos médicos (Obj3.3).
4. **Crear un Asistente Virtual para gestión de citas.** Generar un Asistente Virtual con comunicación efectiva. Este asistente deberá:
 1. Gestionar citas con precisión (agendar, consultar, modificar, cancelar) como tarea principal (Obj4.1).
 2. Ofrecer información médica general, responder preguntas frecuentes sobre el proceso de citas médicas, así como proporcionar orientación básica sobre síntomas y medidas preventivas, estas funciones constituirán su tarea secundaria y mejorará la experiencia del usuario (Obj4.2).
 3. Facilitar la gestión a través de Telegram (Obj4.3).
 4. Incorporar un sistema de guardarrail para evitar solicitudes ajenas a las tareas recomendadas, evitando un uso abusivo del asistente, y recortar los costes asociados de una manera más eficiente (Obj4.4).
5. **Desarrollar un proceso de mejora continua.** Establecer un proceso teórico de mejora continua, donde el modelo predictivo y la información extraída por el Asistente Virtual se retroalimenten periódicamente. El objetivo es mejorar los resultados predictivos del modelo y reducir los costos asociados a inasistencias en la gestión de citas médicas (Obj5.5).

Planificación

La planificación de los recursos económicos se puede ver en el [Anexo 2 – Estimación de recursos económicos](#).

Desarrollo del proyecto

Fase 1. Predicción de Asistencia a Citas Médicas

Elección, recolección y preparación de datos

Un buen modelo predictivo de asistencia a citas médicas requiere un set de datos con las siguientes características:

- Presencia de Variable Target indicando la Asistencia o No Asistencia a la cita médica.
- Presencia de Variables Predictoras diversas y heterogéneas con las que poder predecir si ocurrirá asistencia o no, contra más variables predictoras mejor.
- Gran número de muestras, necesarias para poder generar un modelo predictivo robusto y confiable.

Se hizo una búsqueda por internet y se escogió el siguiente dataset de Kaggle: “Medical Appointment No Shows” ([7](#)), publicado por Jonihoppen.

Dicho Dataset contiene 110.527 muestras (citas médicas) de 62.299 pacientes distintos, de los que se recogen hasta 13 características de las citas y pacientes mencionados, así como la imprescindible característica objetivo de asistencia o no a la propia cita solicitada.

Se encontraron otros conjuntos de datos que fueron descartados por contener menor número de muestras, o peores características predictoras, o simplemente por no disponer de la variable objetivo de Show – No Show.

Escogido el dataset, y antes incluso de realizar el pertinente Análisis Exploratorio de los Datos (EDA), revisamos la nomenclatura de las columnas para corregir errores topográficos en las mismas: Hypertension por Hipertension, Handicap por Handcap, y NoShow por No-show.

Análisis Exploratorio de Datos (EDA)

La primera actividad de peso en el desarrollo de cualquier tarea de ML es realizar un buen Análisis Exploratorio de Datos para extraer toda la información posible de los mismos, y así diseñar la mejor estrategia de datos que le sirva posteriormente al modelo predictivo.

Primero se analiza la información general del dataset: tipos de datos (los cuales se modifican para poder trabajar la información que contiene cada variable), valores únicos, valores nulos, etc.

A continuación, se empieza analizando la variable target u objetivo, y posteriormente se seguirá con el resto de las 13 variables predictoras.

Variable target “NoShow”

Sólo tiene 2 valores posibles: 0 (False) = Show o 1 (True) = No Show. La distribución de valores indica que, como era de esperar, el dataset está fuertemente desbalanceado, revelando un 20.19% de citas sin asistencia.

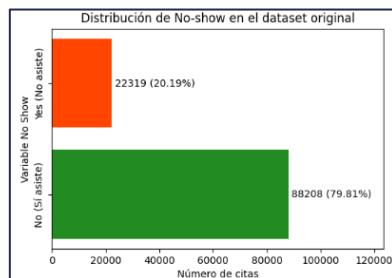


Figura 3. Distribución de citas (asistencias e inasistencias) en el dataset original

Variable predictora: “PatientId”

El dataset consta con información de 62.299 pacientes distintos, identificados con números enteros que van desde el 39217 hasta el 999981631772427. Esto significa que el dataset contiene datos históricos de pacientes que realizan más de una solicitud de cita médica. Esta información puede ser muy relevante a la hora de determinar la probabilidad de asistencia de un paciente, pues en muchos casos ya se tendrá información relevante de su comportamiento en citas previas.

Aun así, tal y como se observa en la siguiente tabla, para el 60.87% de los pacientes sólo se tiene información de una única cita.

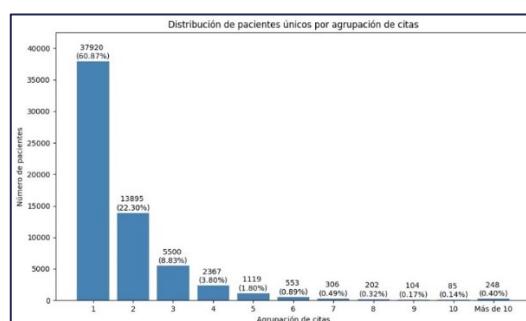


Figura 4. Distribución de pacientes únicos por agrupación de citas médicas

A simple vista no se observa ninguna relación o información de relevancia en el número de identificación del paciente, dato que nos confirmará posteriormente el cálculo de la matriz de correlación entre variables.

Variable predictora: "AppointmentID"

Los valores únicos de esta variable son el total de muestras del dataset: 110.527, identificados con números enteros desde el 5030230 al 5790484.

Tampoco parece haber mayor correlación numérica salvo con la asignación temporal, es decir, que es una identificación numérica correlativa en función de la fecha en la que se solicita la cita. La posible información de utilidad contenida en esa correlación temporal ya está contenida en otras variables más directas, como veremos a continuación.

Variable predictora: "Gender"

Sólo se toman en cuenta dos identificaciones de género: Female and Male, distribuidos en un 65% y 35% de los datos, respectivamente, por lo que esta variable está ligeramente desbalanceada.

Una de las comprobaciones importantes de esta variable, para garantizar coherencia en los datos, es que cada paciente tenga una identificación única e inequívoca del género asignado en cada una de sus correspondientes citas, y así es.

Respecto a la distribución de asistencia según género, las mujeres tienen una proporción mayor de No Shows (20.31%) respecto a los hombres (19,97%).

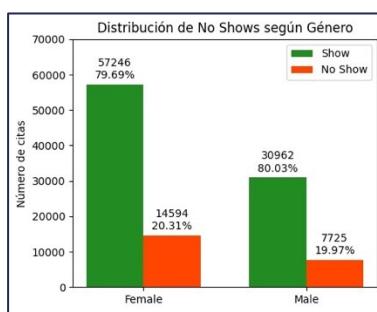


Figura 5. Distribución de citas (asistencias e inasistencias) según el género

Variable predictora: "ScheduledDay"

Esta variable muestra la fecha (día y hora) en la que se solicitó (y programó) la cita médica. Existen 103.549 fechas con día y hora distintos, por lo que hay días en los que se han solicitado citas exactamente en el mismo segundo.

El primer día que se solicitó una cita médica dentro de este conjunto de datos es el 10 de noviembre de 2015, a las 07:13:56, y la última el 8 de junio de 2016 a las 20:07:23.

El siguiente gráfico muestra como la mayoría de las fechas de solicitud de citas se concentra en los meses de abril, mayo y junio de 2016.



Figura 6. Distribución de citas por fecha de programación

Variable predictora: "AppointmentDay"

A diferencia de "ScheduledDay", esta variable sólo guarda información del día en que se tiene la cita médica, sin la hora a la que está convocado cada paciente. Dentro de este dataset, existen 27 días distintos en los que se atienden consultas médicas, desde el 29 de abril al 8 de junio de 2016.

El hecho de contar con 2 variables Datetime relacionadas con la fecha en la que se solicita la consulta y la fecha en la que se atiende, nos permite estudiar posibles dependencias entre el lapso existente entre ambas fechas y la propia asistencia, característica que se estudiará durante el Feature Engineering.

Se planea analizar la correlación entre las asistencias y el día de la semana en que ocurren las citas. No obstante, como se observa en el gráfico de barras, las variaciones en los niveles de asistencia entre los distintos días de la semana son pequeñas.

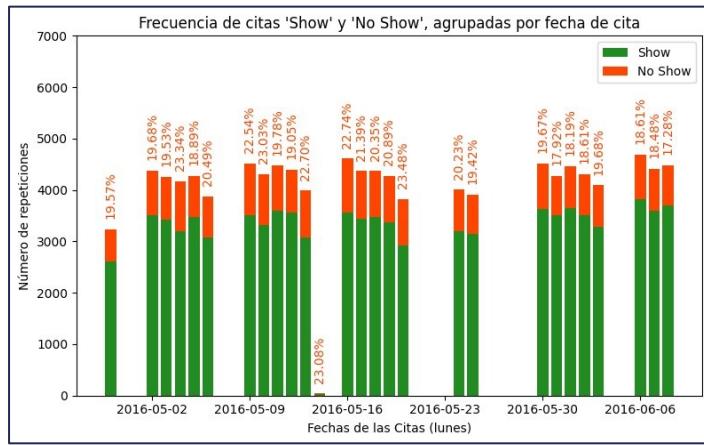


Figura 7. Distribución de citas (asistencias e inasistencias) por fecha de cita

Las fechas indicadas en el gráfico corresponden a todos los lunes de la variable "AppointmentDay", y cada barra representa un día de la semana.

También se observa que el balanceo de los datos para cada día de la semana está bastante equilibrado, salvo para el único sábado del que se tiene constancia (14 mayo), cuyas citas se eliminarán del dataset final para evitar outliers que generen ruido al modelo (**objetivo Obj1.1**).

Variable predictora: "Age"

Respecto a la edad, el dataset comprende 104 edades distintas entre todos los pacientes, desde -1 a 115, según la siguiente distribución agrupadas las edades de 5 en 5:

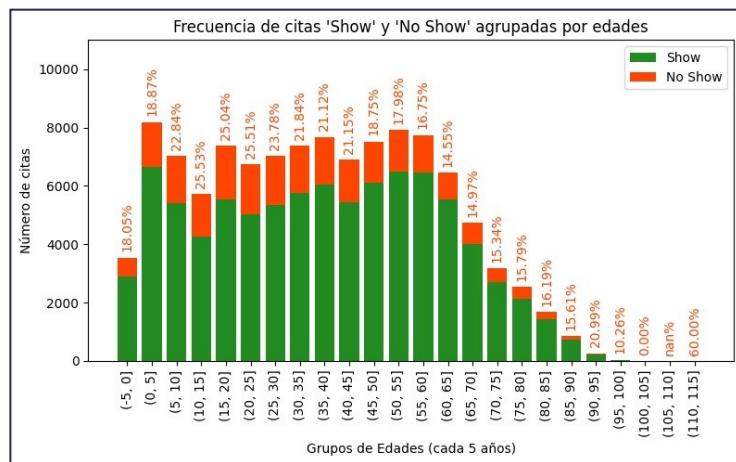


Figura 8. Distribución de citas (asistencias e inasistencias) por rangos de edad

Los pacientes que más faltan a las citas médicas son los comprendidos entre los 5 y los 30 años, así como los mayores de 110, pero de este último grupo hay un único paciente, por lo que no es representativo.

Se observa un severo desbalanceo en los pacientes de mayor de edad, como es natural, pues el número de pacientes es mucho menor. Es importante hacer un estudio más detallado de posibles outliers. Las conclusiones referentes a las edades reflejadas en ambos extremos de los rangos de edad son las siguientes:

- Edades inferiores:
 - -1: Sólo existe una cita para esta edad. Entendemos que es la madre atendiendo una consulta de ginecología.
 - 0, 1, 2, ...: Existen 3539, 2273, 1618, ... citas para estas edades, las cuales no son desdeñables. Obviamente son citas en las que los pacientes (pediátricos) van acompañados por alguno de sus padres.
- Edades superiores:
 - 99: 1 sola cita
 - 100: 4 citas correspondientes a 3 pacientes distintos.
 - 102: 2 citas correspondientes a 2 pacientes distintos.
 - 115: 5 citas correspondientes a un único paciente.

La edad es una información importante para pronosticar la asistencia o no a las citas médicas, por lo que se deja toda la información intacta. Sin embargo, se hacen un par de comprobaciones más para determinar la coherencia y robustez de los datos:

- Se comprueba que ningún paciente recién nacido presenta antecedentes de hipertensión, diabetes o alcoholismo (otras variables predictoras presentes en el dataset): OK.
- Se comprueba que los pacientes sólo tengan una edad asignada: Se detectan 1.168 pacientes con más de una edad asignada, pero cuyas diferencias nunca son mayores de 1 año, por lo que se asume que son pacientes que han cumplido años entre una cita y la siguiente.

Variable predictora: "Neighbourhood"

El dataset contiene información de 81 barrios distintos, todos correspondientes a la ciudad de Vitória, en Brasil.

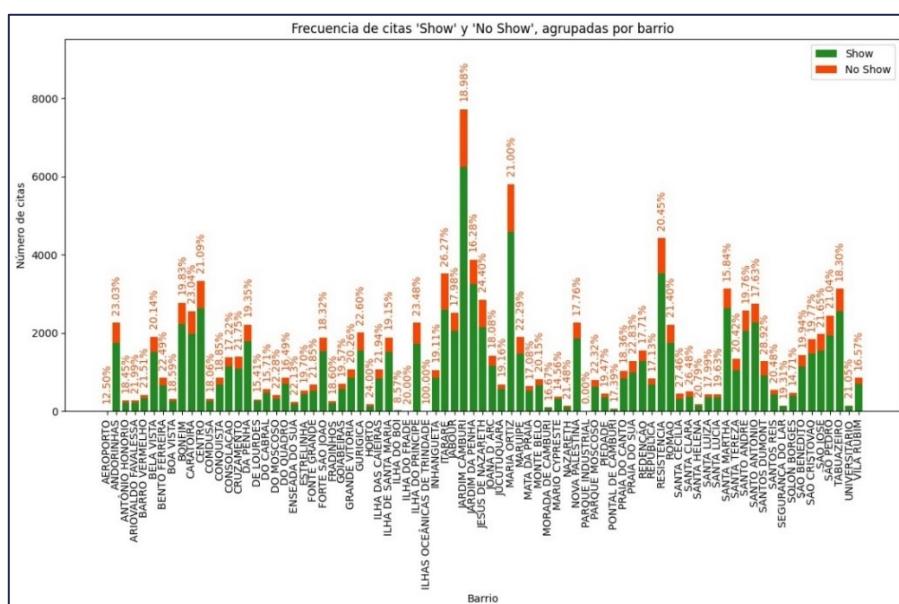


Figura 9. Distribución de citas (asistencias e inasistencias) por barrios

El gráfico muestra un gran desbalanceo de datos en función del barrio considerado, y unas frecuencias de asistencia bastante dispares.

Con el objetivo de mejorar este desbalanceo y tener un modelo que haga unas predicciones más robustas, agruparemos los barrios por clúster de proximidad en el Feature Engineering.

Variable predictora: "Scholarship"

Esta variable contiene información sobre unas becas de ayuda económica que se otorgan en Brasil, siendo una variable binaria indicando si el paciente que solicita la cita dispone de dicha beca o no.

Esta variable está fuertemente desbalanceada, pues sólo un 9.83% de las citas pertenecen a pacientes que disponen de la "Scholarship", los cuales son los que exhiben un mayor porcentaje de No Show.

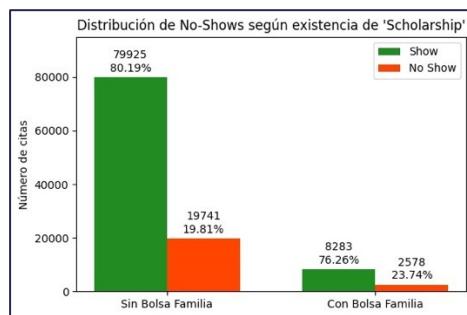


Figura 10. Distribución de citas (asistencias e inasistencias) según ayuda económica

Variable predictora: "Hypertension"

Esta variable también es una variable binaria para indicar si el paciente que acude a la cita tiene hipertensión o no.

Es una variable muy desbalanceada, con un 19.72% de las citas pertenecientes a pacientes que sufren de hipertensión, aunque son los que mejor índice de asistencia tienen.

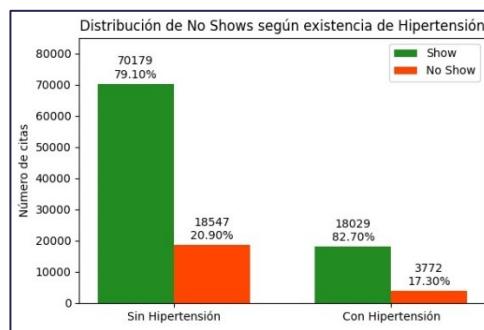


Figura 11. Distribución de citas (asistencias e inasistencias) por hipertensión

Variable predictora: "Diabetes"

Al igual que las dos variables anteriores, ésta también es binaria, indicando si el paciente sufre de diabetes o no.

Esta variable predictora también está fuertemente desbalanceada, con tan sólo un 7.19% de las citas indicando que el paciente solicitante sufre de diabetes, y, al igual que pasaba con los pacientes que sufrían de hipertensión, son los que menos faltan a las citas programadas.

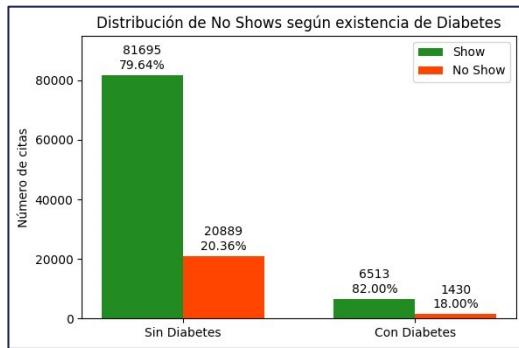


Figura 12. Distribución de citas (asistencias e inasistencias) por diabetes

Variable predictora: "Alcoholism"

Variable predictora binaria que indica si el paciente solicitante de la cita médica es alcohólico o no.

Esta variable está fuertemente desbalanceada, pues tan sólo un 3.04% de las citas del dataset corresponden a pacientes con alcoholismo. Aun así, tampoco es que existan diferencias relevantes en la frecuencia de asistencia a las citas médicas entre un grupo y otro.

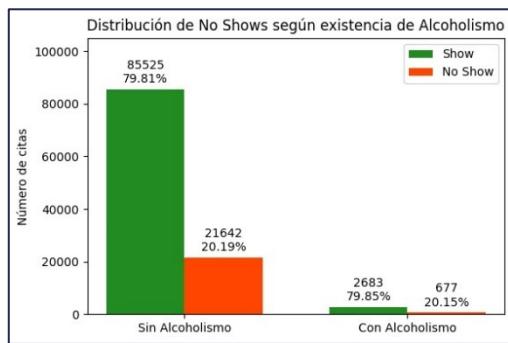


Figura 13. Distribución de citas (asistencias e inasistencias) por alcoholismo

Variable predictora: "Handicap"

Esta variable indica si el paciente que solicita la cita tiene algún grado de discapacidad. Se miden hasta 4 grados de discapacidad, por lo que su valor varía entre 0 (ninguna discapacidad), 1 (grado de discapacidad 1), 2 (grado de discapacidad 2), 3 (grado de discapacidad 3) y 4 (grado de discapacidad 4).

Esta es la variable predictora más desbalanceada de todas: tan sólo el 1.8475% (2.042 citas) corresponde a citas de pacientes con un grado de discapacidad 1, un 0.1656% (183 citas) corresponde citas de pacientes con un grado de discapacidad 2, un 0.0118% (13 citas) corresponden a un grado de discapacidad 3, y un ínfimo 0.0027% corresponden a 3 citas de 3 pacientes distintos con un grado de discapacidad 4.

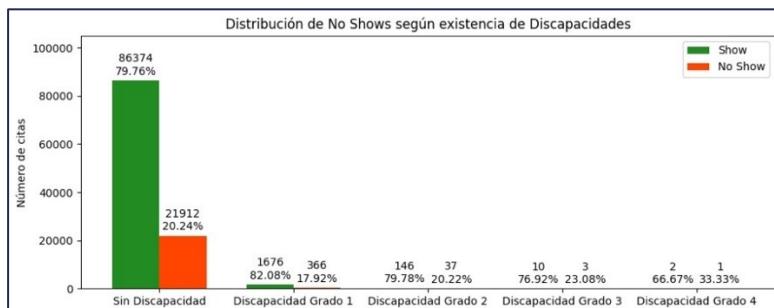


Figura 14. Distribución de citas (asistencias e inasistencias) por grado de discapacidad

Es por eso por lo que todas las citas que indican algún grado de discapacidad en el paciente se consideran outliers. Con tal de mejorar la robustez del modelo se procede a agrupar todos los pacientes con discapacidad a un solo grupo, independientemente del grado de discapacidad médica.

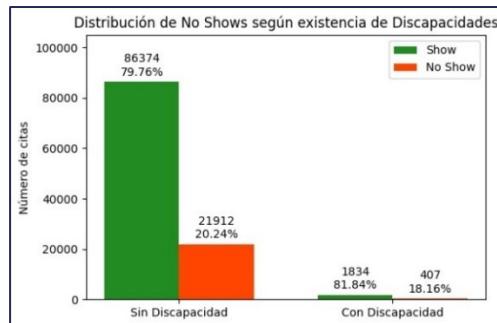


Figura 15. Distribución de citas (asistencias e inasistencias) por discapacidad

Aun así, tal y como se observa en la gráfica anterior, no se consigue solventar el desbalanceo, pues la suma de todas las citas con indicación de algún grado de discapacidad tan sólo representa el 2.0276%.

Es por ello por lo que, vistos también los desbalances existentes en todas las variables predictoras que describen la historia clínica de los pacientes, se trabajarán nuevas características predictoras en la fase de *Featuring Engineering* que aglutinen los pacientes con antecedentes médicos, independientemente de cuál sea.

Variable predictora: "SMS_received"

Esta es la última variable predictora original del dataset, y se trata de una variable binaria que indica si el paciente que asistía a la cita médica recibió [0] o no [1] algún mensaje de texto recordatorio para dicha asistencia.

La variable está desbalanceada pues tan sólo se enviaron mensajes de texto en un 32.10% de las citas contenidas en este dataset.

Llama la atención, y pareciendo “a priori” fuera de toda lógica, los pacientes que recibieron un mensaje de texto recordatorio son los que muestran un porcentaje de No Show mayor.

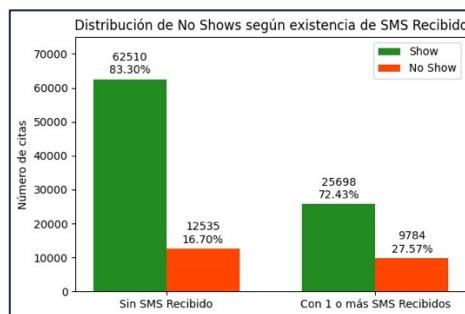


Figura 16. Distribución de citas (asistencias e inasistencias) según SMS recibido

Sin embargo, este comportamiento “fuera de lógica”, como veremos más adelante guarda una correlación encubierta con el lapsus entre la fecha de programación de cita y la fecha de atención a la cita. Los mensajes de texto sólo se envían cuando dicho lapsus supera cierto margen (no se envían mensajes de textos si la cita se programa para el futuro inmediato).

Todo el código utilizado en el EDA se puede consultar en el cuaderno Jupyter Notebook “[01_EDA_Attendance Medical Appointments Brazil.ipynb](#)”.

Feature Engineering

La fase de Feature Engineering en este proyecto se enfoca en transformar y seleccionar las características más relevantes del conjunto de datos para mejorar el rendimiento de los modelos predictivos de IA, específicamente diseñados para predecir la probabilidad de las ausencias (no shows). A continuación, se describen las acciones y transformaciones específicas que se han realizado:

Creación de nuevas características

Para mejorar la capacidad predictiva sobre la asistencia a citas médicas, se han creado nuevas variables predictoras basadas en diferentes aspectos, desde patrones temporales hasta factores geográficos y meteorológicos. Estas variables se han creado para capturar aspectos clave que podrían influir en el comportamiento de los pacientes. A continuación, se describen en detalle estas nuevas características.

Variables temporales (Obj1.2)

Se han creado nuevas variables predictoras basadas en las fechas de solicitud (ScheduledDay) y asistencia a la cita (AppointmentDay). Se parte de la hipótesis de que el día de la semana en que se programa una cita médica puede afectar la probabilidad de asistencia, dado que ciertos días pueden presentar una mayor disponibilidad o disposición por parte de los pacientes para asistir. Así mismo, se plantea que el tiempo transcurrido entre la programación de la cita hasta la fecha real de la misma podría influir en la probabilidad de asistencia.

Nueva variable	Descripción	Variable original
<i>App_DayOfWeek</i>	Día de la semana en que se atiende la cita médica. Valores posibles: 'Friday' 'Wednesday' 'Monday', etc.	AppointmentDay
<i>Time_SchDay_to_AppDay</i>	Tiempo (en segundos) transcurrido entre la fecha de solicitud y la fecha de la cita.	AppointmentDay
<i>Days_since_last_App</i>	Tiempo (en días) transcurrido entre las citas consecutivas por paciente.	AppointmentDay

Variables geográficas (Obj1.2 y Obj1.3)

Se han introducido variables predictoras adicionales basadas en la información geográfica de los barrios, incluyendo su ubicación (8) y la presencia de centros de salud en ellos (9). La hipótesis subyacente sugiere que la proximidad geográfica a los centros médicos y la disponibilidad de atención médica en el barrio pueden influir en la probabilidad de asistencia a las citas médicas.

Nueva variable	Descripción	Variable original
<i>Neigh_Cluster</i>	Etiquetas de clúster asignadas a cada barrio mediante el modelo de K-Means. Esto permite agrupar los barrios según su proximidad geográfica.	LatitudeNeigh LongitudeNeigh
<i>Health_Centre</i>	Identificación binaria que especifica la presencia (1) o ausencia (0) de Centro Médico en cada barrio.	

Se ha utilizado el algoritmo **K-Means**, una técnica de aprendizaje no supervisado, para entender cómo la proximidad geográfica y la disponibilidad de servicios de salud afectan la tasa de ausencia a las citas. El proceso se llevó a cabo de la siguiente manera:

1. **Selección de características.** Para agrupar los barrios, se seleccionaron las coordenadas geográficas (latitud y longitud) de cada barrio. Estas características permiten agrupar los barrios en función de su proximidad geográfica.
2. **Determinación del número de clústers.** Se decidió agrupar los barrios en 12 clústers. Esta decisión se basó en la distribución geográfica y la cantidad total de barrios, buscando un equilibrio entre la granularidad de los grupos y la manejabilidad de los datos.
3. **Aplicación de K-Means.** Utilizando el algoritmo K-Means, se asignó a cada barrio una etiqueta de clúster que indica a qué grupo pertenece.
4. **Ajuste manual de la clusterización.** Tras la clusterización inicial, se realizó un ajuste manual para balancear mejor el número de citas médicas entre los clústers. Este ajuste consistió en mover ciertos barrios entre clústers para asegurar una distribución más equitativa y efectiva de los recursos médicos.

Para facilitar la comprensión y comunicación de los resultados, se creó un [mapa interactivo](#) utilizando [Folium](#). Este mapa muestra los barrios agrupados por clústers, con diferentes colores para cada uno, y señala la ubicación de los centros de salud en rojo. Además, el mapa incluye información sobre el número de citas médicas por barrio, lo que permite identificar de manera clara y accesible la distribución geográfica de los barrios, los recursos de salud disponibles y la carga de trabajo en cada área.

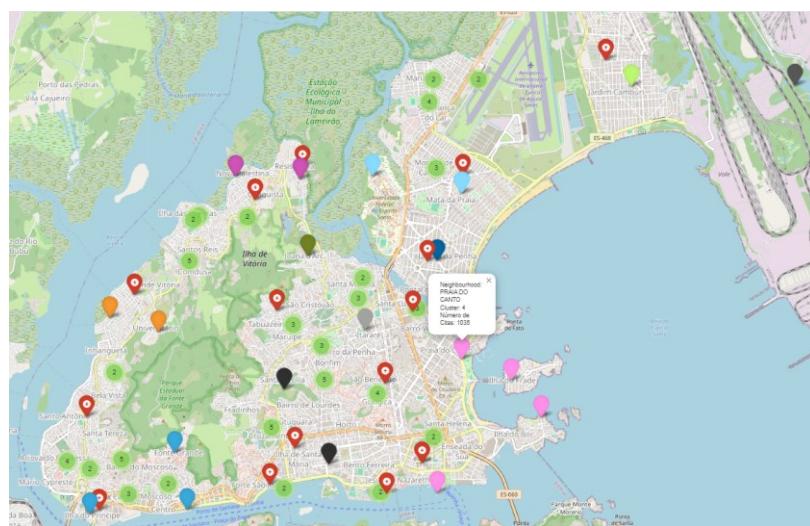


Figura 17. Mapa de distribución de citas médicas por barrio y centros médicos (KMeans)

Variables meteorológicas (Obj1.3)

Otra hipótesis plantea que la asistencia a la cita médica esté fuertemente influenciada por las condiciones meteorológicas del día en cuestión, como la temperatura, la velocidad del viento y la lluvia. Para investigar esta suposición, se procedió a extraer la información meteorológica de la ciudad de Vitoria para el mes de mayo de 2016, obtenida de Weather and Climate (10). Posteriormente, se integraron al conjunto de datos las nuevas variables predictoras: ‘Temperature’, ‘WindSpeed’, ‘Precipitation’.

Variables del historial médico (Obj1.2 y Obj1.3)

Bajo la premisa de que la combinación de las condiciones médicas de un paciente puede tener relevancia en la predicción de la asistencia a las citas médicas, se han introducido variables adicionales basadas en el historial médico de los pacientes.

Nueva variable	Descripción	Variable original
<i>Number_Health_Conds</i>	Número total de condiciones médicas que tiene cada paciente. Valores posibles: 0, 1, 2, 3, 4.	Hypertension Diabetes Alcoholism Handicap
<i>Presence_Health_Conds</i>	Identificación binaria que especifica la presencia (1) o ausencia (0) de condiciones médicas por paciente.	<i>Number_Health_Conds</i>

Además, para determinar la probabilidad de asistencia del paciente, se han considerado las siguientes estadísticas:

Nueva variable	Descripción
<i>Prior_Apps_byPatient</i>	Número de citas previas de cada paciente.
<i>Prior_NoShows_byPatient</i>	Indica la cantidad de veces que un paciente no asistió a citas previas.
<i>Prob_NoShow_byPatient</i>	Porcentaje de no asistencia del paciente, basado en su historial de citas.

Es importante señalar que el porcentaje de no asistencia se inicializa para cada paciente con el valor promedio de no asistencia en todo el conjunto de datos. Posteriormente, este valor se actualiza según el historial específico de cada paciente a medida que tiene citas médicas.

Transformación de tipos de datos

Durante el proceso de preparación de los datos, se realizaron varias transformaciones para garantizar la coherencia y la adecuación de los tipos de datos. A continuación, se detallan las principales transformaciones llevadas a cabo:

Conversiones de tipo de datos numéricos. Se ajustaron los tipos de datos numéricos, como enteros o flotantes, para garantizar la consistencia en las operaciones matemáticas y el análisis estadístico. Esto incluyó la conversión de variables como el identificador del paciente (PatientId) y el género (Gender).

Manipulación de fechas y horas. Las variables relacionadas con fechas y horas, como las fechas de programación (ScheduledDay) y las fechas de las citas médicas (AppointmentDay), se convirtieron al formato de fecha y hora adecuado para facilitar su manipulación y análisis.

Codificación de variables categóricas. Se utilizó LabelEncoder para convertir variables categóricas, como el género (Gender), día de la semana de la cita (App_DayOfWeek) y el barrio (Neighbourhood), en valores numéricos. Esto se realizó para permitir el procesamiento de algoritmos de aprendizaje automático que requieren datos numéricos como entrada.

Además, la **variable target “No Show”** fue transformada a tipo booleano para facilitar su manipulación y análisis durante el modelado predictivo: 0 (sí asiste), 1 (no asiste).

Tratamiento de valores atípicos y missing values (Obj1.1)

Durante el análisis de datos, se identificaron valores atípicos y algunos registros que requirieron un tratamiento especial. A continuación, se detallan las acciones tomadas:

Condición de discapacidad. La variable “Handicap” originalmente incluía valores de 0 a 4, indicando niveles de discapacidad. Sin embargo, el análisis exploratorio de datos (EDA) reveló que los niveles superiores a 0 no eran relevantes. Por lo tanto, se simplificó la variable para que solo indique si un paciente tiene discapacidad (1) o no (0). Ver Figura 15.

Eliminación de citas programadas para los sábados. La variable “App_DayOfWeek” mostró que solo había 39 citas programadas para los sábados. Para evitar posibles sesgos debido al pequeño tamaño de esta muestra y reducir el ruido en el modelo, se decidió eliminar estas citas del conjunto de datos.

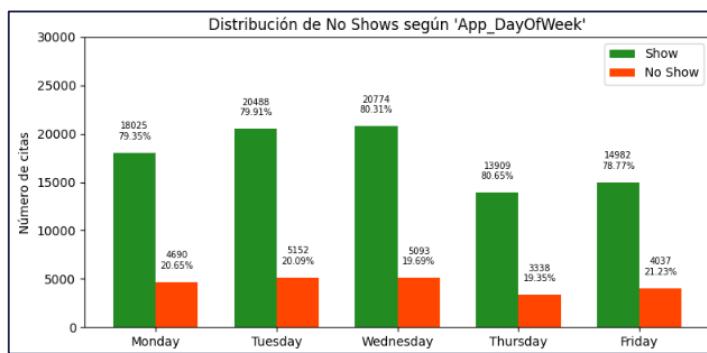


Figura 18. Distribución de citas (asistencias e inasistencias) según el día programado

Gestión de valores negativos en el tiempo entre programación y cita. Se identificaron 5 registros en la variable “Time_SchDay_to_AppDay” con valores negativos, indicando que la fecha de programación era posterior a la fecha de la cita. Estos casos se corrigieron asignando un valor de 0, asumiendo que el paciente programó la cita el mismo día de la consulta.

PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	App_DayOfWeek	Time_SchDay_to_AppDay	Age	Neighbourhood	Scholarship	Hypertension	Diabetes	Alcoholism	Handicap	SMS_received	NoShow
72362	3787481966821	M	2016-05-04 06:50:57	2016-05-03	Tuesday	-111057	7	TABUAZEIRO	0	0	0	0	0	0	True
27033	7839272661752	M	2016-05-10 10:51:53	2016-05-09	Monday	-125513	38	RESISTÊNCIA	0	0	0	0	1	0	True
55226	7896283967868	F	2016-05-18 14:50:41	2016-05-17	Tuesday	-139841	19	SANTO ANTÔNIO	0	0	0	0	1	0	True
64175	24252258389979	F	2016-05-05 13:43:58	2016-05-04	Wednesday	-135838	22	CONSOLAÇÃO	0	0	0	0	0	0	True
71533	99823158161212	F	2016-05-11 13:49:20	2016-05-05	Thursday	-568160	81	SANTO ANTÔNIO	0	0	0	0	0	0	True

Figura 19. Paciente con fecha de programación posterior a la fecha de la cita

```
[ ] # Corrección a 0's de todos los 'Time_SchDay_to_AppDay' negativos:
med_app_FE['Time_SchDay_to_AppDay'][med_app_FE['Time_SchDay_to_AppDay'] <= 0] = 0
```

Figura 20. Modificación de los valores negativos de “Time_SchDay_to_AppDay” a 0

Asignación de valor -1 a pacientes sin citas previas. En la variable “Days_since_last_App”, los pacientes sin citas previas recibieron un valor de -1, indicando la falta de datos previos para calcular el tiempo transcurrido.

```
[ ] # Cálculo e inserción de la variable del tiempo entre citas de un mismo paciente
med_app_FE.insert(loc = 7,
                   column = 'Days_since_last_App',
                   value = med_app_FE.groupby('PatientId')['AppointmentDay'].diff().dt.days)

# Asignación -1 a todos las citas de pacientes que no han tenido citas previas
med_app_FE['Days_since_last_App'].fillna(-1, inplace=True)
```

Figura 21. Asignación del valor -1 a pacientes sin citas previas en “Days_since_last_App”

Ajuste de clústeres en “Neigh_Cluster”. Se detectaron clústeres con solo 1 o 2 muestras (específicamente los clústeres 1 y 5). Estos clústeres se eliminaron y se reorganizaron los restantes, reduciendo el número de clústeres de 12 a 10 para eliminar los menos representativos.

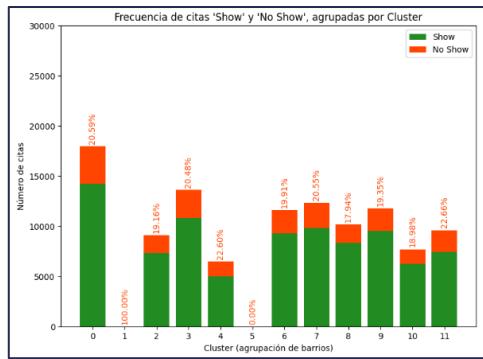


Figura 22. Distribución de citas (asistencias e inasistencias) agrupadas por cluster

```
[ ] # Eliminación de los 2 clusters con una o dos cita:
indexes = med_app_FE.loc[(med_app_FE['Neigh_Cluster'] == 1) | \
                           (med_app_FE['Neigh_Cluster'] == 5)]
                           ].index.tolist()

med_app_FE.drop(indexes, inplace = True)

# Corremos todos los valores de los cluster del 0 al 9:
for i in range(2, 12):
    if i < 6:
        med_app_FE.loc[med_app_FE['Neigh_Cluster'] == i, 'Neigh_Cluster'] = i-1
    else:
        med_app_FE.loc[med_app_FE['Neigh_Cluster'] == i, 'Neigh_Cluster'] = i-2
```

Figura 23. Eliminación de cluster con bajo número de citas médicas

Es importante señalar que en el dataset original no se encontraron valores faltantes. Estas medidas aseguran la integridad y coherencia de los datos utilizados para el análisis posterior.

Validación de características

El análisis de correlación se utilizó para identificar relaciones lineales entre las características y la variable objetivo, así como entre las propias características. Esto incluye:

Matriz de correlación. Se calculó la matriz de correlación de Pearson para todas las características numéricas, lo que permitió identificar características redundantes que podrían ser eliminadas o combinadas.

Correlación con la variable objetivo. Se analizaron las correlaciones entre cada característica y la variable objetivo ('NoShow') para identificar las características con mayor poder predictivo.

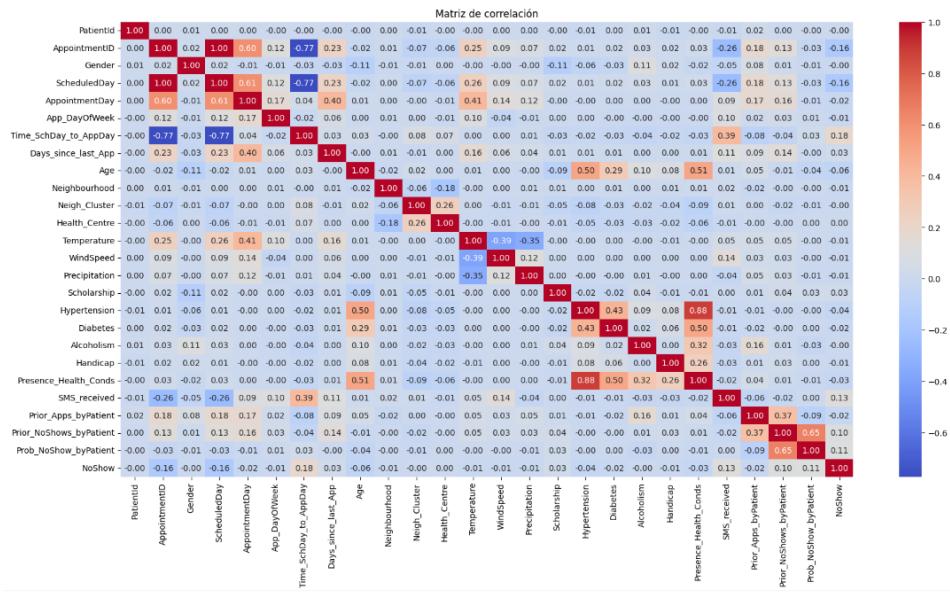


Figura 24. Matriz de correlación sin reducción de dimensionalidad

Selección de características

En esta sección, se presenta una comparación detallada entre la estructura del dataset original y el dataset resultante después de aplicar técnicas de ingeniería de características (Feature Engineering).

Dataset Original

El dataset inicial contiene una serie de variables básicas y diversas para la gestión de citas médicas. Estas variables proporcionan una base inicial que, aunque útil, requiere refinamiento para mejorar su capacidad predictiva.

Dataset Tratado

Tras aplicar técnicas de feature engineering, el dataset original se modificó. Se añadieron nuevas variables predictoras y se ajustaron los tipos de datos de varias de las variables originales. En la imagen incluida, se destacan en color rojo las modificaciones y las nuevas variables añadidas. Esta transformación es fundamental para aumentar la capacidad del modelo de IA para identificar patrones y tendencias relacionadas con la asistencia a citas médicas.

Dataset Final

Finalmente, se realizó una selección exhaustiva de características para crear el dataset final. Este conjunto de datos, que se utilizará para la normalización, escalado, reducción de dimensionalidad y particionamiento, incluye solo las variables más relevantes y útiles para la predicción de asistencia a citas médicas. En la imagen, las variables eliminadas se muestran en gris, mientras que las variables seleccionadas se destacan en negrita.

DATASET ORIGINAL

PatientId	float64
AppointmentID	int64
Gender	object
ScheduledDay	object
AppointmentDay	object
Age	int64
Scholarship	int64
Hypertension	int64
Diabetes	int64
Alcoholism	int64
Handicap	int64
SMS_received	int64
NoShow	object

DATASET TRATADO

PatientId	int64
AppointmentID	int64
Gender	int32
ScheduledDay	datetime64[ns]
AppointmentDay	datetime64[ns]
App_DayOfWeek	int32
Time_SchDay_to_AppDay	int64
Days_since_Last_App	int64
Age	int64
Neighbourhood	int32
Neigh_Cluster	int64
Health_Centre	int64
Temperature	float64
WindSpeed	float64
Precipitation	float64
Scholarship	int64
Hypertension	int64
Diabetes	int64
Alcoholism	int64
Handicap	int64
Presence_Health_Conds	int64
SMS_received	int64
Prior_Apps_byPatient	int64
Prior_NoShows_byPatient	int64
Prob_NoShow_byPatient	float64
NoShow	bool

DATASET FINAL

1 Gender	int32
2 ScheduledDay	datetime64[ns]
3 AppointmentDay	datetime64[ns]
4 App_DayOfWeek	int32
5 Time_SchDay_to_AppDay	int64
6 Days_since_Last_App	int64
7 Age	int64
8 Neighbourhood	int32
9 Neigh_Cluster	int64
10 Health_Centre	int64
11 Temperature	float64
12 WindSpeed	float64
13 Precipitation	float64
14 Scholarship	int64
15 Hypertension	int64
16 Diabetes	int64
17 Alcoholism	int64
18 Handicap	int64
19 Presence_Health_Conds	int64
20 SMS_received	int64
21 Prior_Apps_byPatient	int64
22 Prior_NoShows_byPatient	int64
23 Prob_NoShow_byPatient	float64
24 NoShow	bool

Figura 25. Transformaciones del dataset

Iteraciones y refinamiento

Durante el proceso de Feature Engineering para el proyecto de predicción de asistencia a citas médicas, se llevaron a cabo varias iteraciones y refinamientos para optimizar las características utilizadas en el modelo predictivo. Estas iteraciones se basaron en los hallazgos del análisis exploratorio de datos, así como en la retroalimentación de la validación del modelo y los cambios en los requisitos del proyecto.

Iteraciones

- Se realizaron iteraciones en la selección y creación de características en respuesta a los patrones observados durante el análisis exploratorio de datos. Por ejemplo, se exploraron diferentes formas de representar el historial médico de los pacientes para capturar de manera efectiva su impacto en la asistencia a las citas.
- Se llevaron a cabo ajustes en las características temporales, como el tiempo transcurrido entre la programación de la cita y la fecha real de la misma, para capturar con mayor precisión la influencia de los factores temporales en la probabilidad de asistencia.
- Se exploraron técnicas avanzadas de transformación de variables geográficas, como la segmentación espacial, para capturar la influencia de la ubicación geográfica en la asistencia a las citas médicas.

Refinamiento

- Se refinaron las características seleccionadas en función de su capacidad predictiva y su interpretabilidad. Por ejemplo, se realizaron ajustes en las características basadas en el historial médico de los pacientes para equilibrar la representación de diferentes condiciones médicas.
- Se evaluaron y refinaron las técnicas de imputación de valores faltantes y manejo de valores atípicos para mejorar la calidad de los datos y reducir el impacto de datos erróneos en el modelo.

Normalización y escalado de características

La estandarización y el escalado de características son procesos fundamentales en el preprocessamiento de datos para muchos algoritmos de aprendizaje automático. En nuestro proyecto de gestión de citas médicas, estos pasos son esenciales para asegurar que las características de entrada estén en una escala comparable, facilitando así la convergencia de los modelos y mejorando su capacidad predictiva.

Utilizamos la clase **StandardScaler** de la biblioteca scikit-learn para estandarizar nuestras bases de datos. Este proceso implica ajustar el escalador usando solo los datos de entrenamiento, permitiendo que el escalador "aprenda" de estos datos. Luego, aplicamos la misma transformación a los datos de prueba utilizando el escalador ajustado, garantizando que la distribución de cada característica tenga una media de 0 y una desviación estándar de 1.

```
# Se crea una instancia de StandardScaler
scaler = StandardScaler()

# Se ajusta el escalador a los datos de entrenamiento y se transforman
X_train_scaled = scaler.fit_transform(X_train_set)
# Se transforman los datos de test utilizando el mismo escalador que se ajustó a los datos de entrenamiento
X_test_scaled = scaler.transform(X_test_set)
```

Figura 26. Estandarización con StandardScaler

En la gestión de citas médicas, la estandarización es crucial debido a la diversidad de las características consideradas, como la edad del paciente y el tiempo entre la solicitud y la cita. Para la edad de los pacientes, originalmente de -1 a 115 años, la estandarización ha ajustado el rango a valores entre -1 y 3. Esto asegura que las variaciones de las variables no afecten negativamente el rendimiento de los modelos, manteniendo la integridad de los datos originales.

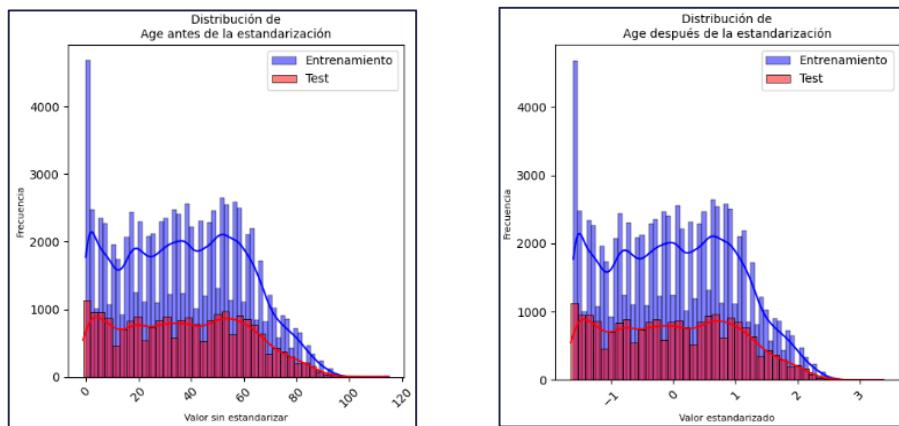


Figura 27. Distribución de citas por edad sin estandarizar (izquierda) y estandarizado (derecha)

Para abordar el desbalanceo de clases en nuestro conjunto de datos de entrenamiento, hemos aplicado dos técnicas de **Data Augmentation** de manera independiente: SMOTE-ENN y ADASYN.

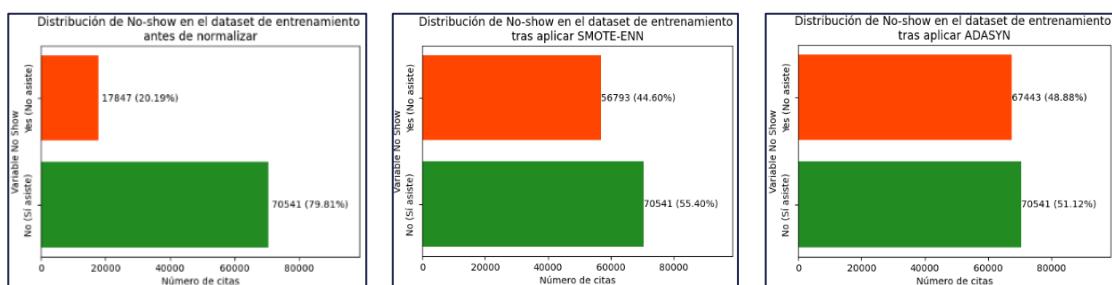


Figura 28. Dataset Entrenamiento - Distribución de citas (sin balancear, SMOTE-ENN, ADASYN)

SMOTE-ENN

Implementamos la técnica SMOTE-ENN mediante el uso de Pipeline de scikit-learn. SMOTE (*Synthetic Minority Over-sampling Technique*) genera nuevas muestras sintéticas de la clase minoritaria, mientras que ENN (*Edited Nearest Neighbours*) elimina instancias ruidosas, logrando así un equilibrio de clases en este conjunto de datos como se visualiza en la imagen.

```
[ ] # Se crea una instancia de SMOTE y ENN
smote = SMOTE(sampling_strategy = 'minority',
               random_state = 42)
enn = EditedNearestNeighbours(sampling_strategy = 'not minority',
                               kind_sel = 'all',
                               n_neighbors = 2)

# Se crea una instancia de la clase Pipeline
pipeline = Pipeline([('smote', smote), ('enn', enn)])

# Se aplica el pipeline a los datos
X_train_resampled, y_train_resampled = pipeline.fit_resample(X_train_scaled, y_train_set)
```

Figura 29. Aplicación de SMOTE-ENN

ADASYN

En otra copia independiente de los datos de entrenamiento, aplicamos ADASYN (Adaptive Synthetic Sampling Approach). ADASYN es una técnica que no sólo genera nuevas muestras sintéticas para la clase minoritaria, sino que lo hace de manera adaptativa. Esto significa que ADASYN prioriza la generación de nuevas muestras en las áreas del espacio de características donde los datos son más escasos y difíciles de clasificar. Al enfocarse en las casuísticas con menor densidad de datos, se espera que ADASYN mejore la capacidad del modelo para manejar casos difíciles y, por lo tanto, aumente la precisión del modelo en la clasificación de la clase minoritaria.

```
[ ] # Crear una instancia de ADASYN
adasyn = ADASYN(sampling_strategy='minority', random_state=42)

# Aplicar ADASYN a los datos de entrenamiento
X_train_resampled, y_train_resampled = adasyn.fit_resample(X_train_scaled, y_train_set)
```

Figura 30. Aplicación de ADASYN.

Reducción de dimensionalidad

La reducción de dimensionalidad es una técnica fundamental para simplificar el conjunto de datos y mejorar la eficiencia de los modelos de inteligencia artificial sin perder información importante. Para este propósito, se ha implementado el **Análisis de Componentes Principales (PCA)**, una técnica ampliamente utilizada en aprendizaje automático.

Con el PCA, nuestro objetivo era conservar al menos el 95% de la varianza total mientras reducímos la dimensionalidad de nuestro conjunto de datos.

```
[ ] # Aplicar PCA para reducir la dimensionalidad mientras se conserva la mayor cantidad de varianza posible
pca = PCA(n_components = 0.95,
           svd_solver = 'full')

# Ajustar y transformar los datos de entrenamiento
X_train_pca = pca.fit_transform(X_train_resampled)
X_test_pca = pca.transform(X_test_scaled)
```

Figura 31. Aplicación de PCA

Sin embargo, después de aplicar el PCA, observamos que, de las 19 variables predictoras originales, solo se seleccionaron 16 componentes principales. A pesar de ser un número menor de variables, estos componentes lograron explicar el 96.66% de la varianza total de los datos, proporcionando una representación efectiva de la información contenida en el conjunto de datos original.

```
Número de componentes: 16
Varianza explicada por cada componente: [0.13707614 0.12212461 0.09235221 0.0689055  0.06253879 0.05901496
0.05718571 0.05134638 0.05082864 0.05010441 0.04323952 0.03925022
0.03778197 0.03715579 0.03256729 0.02512141]
Varianza total explicada: 0.9665935231029232
```

Figura 32. Resultado de aplicación de PCA

Dada la posibilidad de que la reducción en el número de variables conlleve a una pérdida de información, consideramos que era más prudente mantener todas las variables originales en nuestros modelos. Esta decisión se basó en la observación de que el PCA eliminó sólo 3 variables, y que esta pérdida no justificaba la posible reducción en la calidad de nuestras predicciones, especialmente considerando que ya estábamos conservando una varianza explicada del 95%.

Entrenamiento de modelos

Particionamiento de datos

Dado que no se disponía de un conjunto de datos de pruebas, se procedió a generar uno mediante el particionamiento de los datos disponibles. Este proceso se realizó con el propósito de evaluar el rendimiento de los modelos predictivos en datos no vistos durante el entrenamiento. Previamente, se decidió crear conjuntos de datos de acuerdo a hipótesis sobre los que posteriormente se aplicaría el particionamiento.

Hipótesis para la creación de conjuntos de datos:

- **P1-ALL: Mantenimiento de la totalidad del conjunto de datos.** Bajo la premisa de que la retención de la totalidad de los datos maximiza la información disponible y potencialmente mejora la capacidad predictiva, se mantuvo el conjunto de datos original sin ningún tipo de filtrado.
- **P2-NOCONDITIONS: Pacientes sin condiciones médicas.** Se creó un conjunto de datos que incluyó solo a los pacientes del conjunto original que no tenían condiciones médicas registradas. Esto se basó en la hipótesis subyacente de que los pacientes sin condiciones médicas pueden tener una mayor probabilidad de inasistencia. Al generar este dataset, se eliminaron las variables 'Hypertension', 'Diabetes', 'Alcoholism', 'Handicap', 'Presence_Health_Conds'.
- **P3-AGE: Pacientes de edad entre 5 y 30 años.** Se generó un conjunto de datos que contenía únicamente a los pacientes del conjunto original cuya edad se encontraba entre 5 y 30 años. Esto se basó en la hipótesis subyacente de que este grupo demográfico tiende a faltar más a las citas médicas.
- **P4-TIME: Pacientes con citas programadas para otro día.** Se formó un conjunto de datos que incluyó exclusivamente a los pacientes del conjunto original cuyas citas estaban programadas para un día distinto al de su solicitud. Esto se sustentó en la hipótesis de que existía una correlación significativa entre este grupo de pacientes y la inasistencia.
- **P5-HEALTHCENTRE: Pacientes de barrios con centro médico.** Se estableció un conjunto de datos que contenía únicamente a los pacientes del conjunto original de barrios donde existía un centro médico. Esto se basó en la hipótesis de que los pacientes que estaban en barrios con centros médicos podían tener una menor sensación de urgencia o responsabilidad al tener acceso fácil a atención médica.
- **Pacientes segmentados por grupos de edad.** Se crearon conjuntos de datos específicos para diferentes grupos de edad, basados en la hipótesis de que la edad podía influir en los patrones de asistencia a citas médicas:
 - **P6A-KIDS.** Niños: Menores de 12 años
 - **P6B-ADOLESCENTS.** Adolescentes: 13 - 18 años
 - **P6C-YOUNGADULTS.** Jóvenes adultos: 19 - 35 años
 - **P6D-ADULTS.** Adultos: 36 - 64 años
 - **P6E-OLDERADULTS.** Adultos mayores: 65 años en adelante

Procedimiento de particionamiento de datos:

Una vez formuladas las hipótesis y generados los conjuntos de datos correspondientes, se procedió al particionamiento de los datos para entrenamiento y prueba. Este proceso se llevó a cabo con el objetivo de evaluar y validar la eficacia de los modelos de aprendizaje automático en datos no vistos durante el entrenamiento. El procedimiento se detalló a continuación:

1. **Cálculo de variables relevantes.** Para cada hipótesis, se calculó el número de pacientes en el conjunto de datos (NUMBER_PATIENTS), el número de citas correspondientes a no asistencia (NUMBER_NO_SHOWS) y el número total de muestras en el conjunto de datos (NUMBER_SAMPLES).
2. **Determinación del tamaño del conjunto de prueba.** Se determinó el tamaño del conjunto de prueba multiplicando el número total de muestras por un porcentaje predefinido, usualmente el 20%. Esto garantizó una división adecuada entre los conjuntos de entrenamiento y prueba.

```
[ ] NUMBER_Apps_inTest = int(NUMBER_SAMPLES * 0.20)
print(f"Para alcanzar el 20% del muestreo en nuestro set de pruebas\n \
necesitamos {NUMBER_Apps_inTest} citas de pacientes distintos, es decir,\n \
muestras de un {NUMBER_Apps_inTest / NUMBER_PATIENTS:.2%} de los pacientes.")

→ Para alcanzar el 20% del muestreo en nuestro set de pruebas
necesitamos 22097 citas de pacientes distintos, es decir,
muestras de un 35.48% de los pacientes.
```

Figura 33. Determinación del tamaño del conjunto de prueba

3. **Creación de subconjuntos de datos.** Se creó un subconjunto de datos que contenía únicamente la última cita de cada paciente, ordenado según la fecha de la cita, para cada hipótesis generada. Esto aseguró que cada paciente estuviera representado por una única muestra en el conjunto de prueba.
4. **Determinación de la proporción de no asistencia.** Se calculó la cantidad de pacientes que no asistieron y que asistieron, necesarios para mantener la misma proporción en el conjunto de prueba como en el conjunto de datos completo, para cada hipótesis. Esto aseguró que la distribución de asistencia y no asistencia se mantuviera en el conjunto de prueba.

```
# Cálculo del número de citas NoShow = True que se requieren para mantener la estratificación de Clases
NUMBER_NoShows_inTest = int(NUMBER_Apps_inTest * (NUMBER_NO_SHOWS / NUMBER_SAMPLES))
NUMBER_Shows_inTest = int(NUMBER_Apps_inTest - NUMBER_NoShows_inTest)
print(f"Se requieren {NUMBER_NoShows_inTest} pacientes con 'NoShow' = True en el Set de Prueba, y\n \
{NUMBER_Shows_inTest} pacientes con 'NoShow' = False.\n")
```

Figura 34. Determinación de la proporción de no asistencia

5. **Selección de citas para el conjunto de prueba.** Se seleccionaron las últimas citas de pacientes que no asistieron y asistieron según el número calculado anteriormente, para cada hipótesis. Estas citas se utilizaron para formar el conjunto de datos de prueba final.
6. **Filtrado del conjunto de datos original.** Después de haber seleccionado las citas necesarias para el conjunto de prueba, se utilizaron esas citas para filtrar el conjunto de datos original y crear tanto el conjunto de prueba como el conjunto de entrenamiento definitivos, para cada hipótesis. Esto garantizó que cada conjunto de prueba final contuviera exactamente las citas necesarias, mientras que cada conjunto de entrenamiento final contuviera todas las demás citas que no estaban en el conjunto de prueba.

Entrenamiento de modelos

Se procedió a entrenar, para cada una de las distintas bases de datos, cuatro modelos: dos propios de ML, regresión logística y árbol de decisión, en ambos casos utilizando la librería Sklearn de Python; y dos redes neuronales, una red neuronal densa (NeuralNet) y un modelo de atención secuencial (TabNet), utilizando la librería PyTorch, entrenados utilizando GPU (Graphics Processing Unit). Se desestimó la utilización de otros modelos de ML de uso intensivo de CPU (Central Processing Unit) como Support Vector Machine (SVM) y modelos ensamblados como Random Forest, AdaBoost y Stacking

debido a las limitaciones del hardware que disponíamos. A continuación, se detalla el entrenamiento de cada modelo:

Para cada modelo de Sklearn, se entrenaron tres instancias de **GridSearchCV** (Obj2.1), cada una optimizada para una métrica de rendimiento (*scoring*) diferente: **Accuracy** (Exactitud), **Precision** (Precisión) y **Recall** (Sensibilidad).

Regresión logística.

Se procedió a realizar una búsqueda de mejores hiperparámetros utilizando búsqueda en Cuadrícula (Grid Search): Se utilizó GridSearchCV para explorar un espacio de hiperparámetros, probando distintos valores de 'C', algoritmos de optimización ('solver') y el número máximo de iteraciones ('max_iter'). El objetivo fue encontrar la combinación óptima de estos parámetros que minimizara la función de pérdida y mejorara el rendimiento del modelo de regresión logística.

```
# Se define el rango de hiperparámetros para la búsqueda en cuadrícula
param_grid = {
    'C': [0.01, 0.1, 1, 10, 100],
    'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],
    'max_iter': [100, 200, 300, 500, 1000]
}
```

Figura 35. Hiperparámetros en regresión logística

Posteriormente se realizó validación Cruzada (K-Fold Cross-Validation): Se utilizó un KFold con 5 particiones (n_splits=5) y se barajaron los datos (shuffle=True) para asegurar una evaluación robusta del modelo. Durante cada iteración del K-Fold, se ajustó el modelo y se evaluó el rendimiento en un conjunto de validación. El modelo con la mejor puntuación de validación se seleccionó como el mejor, se guardó y se utilizó en el conjunto de prueba.

Árbol de decisión.

Al igual que en la regresión logística, se procedió a realizar una búsqueda de mejores hiperparámetros utilizando búsqueda en Cuadrícula (Grid Search): Se utilizó GridSearchCV para explorar un espacio de hiperparámetros en un modelo de árbol de decisión. En este caso, se evaluaron los criterios de división ('criterion'), la profundidad máxima del árbol ('max_depth'), el número mínimo de muestras para dividir un nodo ('min_samples_split') y el número mínimo de muestras en una hoja ('min_samples_leaf'). El objetivo fue optimizar la capacidad del modelo para generalizar sin caer en el sobreajuste.

```
# Se define el rango de hiperparámetros para la búsqueda en cuadrícula
param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [10, 15, 20],
    'min_samples_split': [2, 3, 5],
    'min_samples_leaf': [2, 3, 4]
}
```

Figura 36. Hiperparámetros en árbol de decisión

Se repitió el mismo proceso de validación Cruzada (K-Fold Cross-Validation): Se utilizó un KFold con 5 particiones (n_splits=5), se barajaron los datos (shuffle=True) y se evaluó el rendimiento en un conjunto de validación. El modelo con la mejor puntuación se seleccionó, se guardó y se utilizó en el conjunto de prueba.

Redes neuronales: NeuralNet

NeuralNet es una red neuronal densa, lo que significa que cada capa está conectada directamente a todas las capas posteriores. Se define una clase NeuralNet que hereda de nn.Module. La red consta de:

- **Capas lineales (nn.Linear):** Estas capas son responsables de las transformaciones lineales de los datos de entrada, aprendiendo pesos y sesgos que se ajustan durante el entrenamiento.

- **Normalización por lotes (nn.BatchNorm1d):** Reduce el desplazamiento interno de covariables, haciendo el entrenamiento más eficiente.
- **Funciones de activación ReLU (nn.ReLU):** Introducen no linealidades en el modelo, permitiendo que la red neuronal aprenda relaciones complejas en los datos.
- **Función de activación Sigmoid (nn.Sigmoid):** Se utiliza en la capa de salida para transformar las salidas en probabilidades, es útil ya que es un problema de clasificación binaria, y fundamental para la Fase 2 del proyecto.
- **Capa de Dropout (nn.Dropout):** desactiva aleatoriamente algunas neuronas durante el entrenamiento, ayuda a prevenir el sobreajuste.

Validación: Se utilizó StratifiedKFold con 5 particiones (`n_splits=5`) y se barajaron los datos (`shuffle=True`) para asegurar una evaluación robusta del modelo y se dividió el conjunto de datos en pliegues estratificados para mantener la proporción de clases en cada pliegue. Para cada pliegue, se crearon tensores de entrenamiento y validación. Se utilizaron DataLoader para cargar los datos en mini-lotes (`batch_size=64`), lo que facilitó el entrenamiento por lotes. En cada época, se entrenó el modelo, se calculó la pérdida de entrenamiento y validación, y se ajustó el optimizador.

Entrenamiento: Se establecieron los parámetros de entrenamiento como:

- Número de épocas (`num_epochs=100`).
- Criterio de Pérdida (`nn.BCELoss`), Optimizador (`optim.Adam`) con `lr=1e-4` y `weight_decay=1e-5`, útil en problemas de clasificación binaria.
- Scheduler para ajustar la tasa de aprendizaje `ReduceLROnPlateau` con `factor=0.5` y `patience=3`. Esto permitió ajustar dinámicamente la tasa de aprendizaje, mejorando la convergencia del modelo.
- Se implementó el Early Stopping con `patience=3` para detener el entrenamiento si no había mejora en la pérdida de validación, evitando así el sobreajuste.

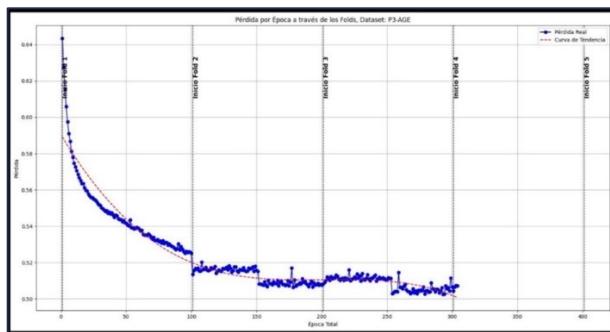


Figura 37. Pérdida por época en red neural

Se calcularon las pérdidas de entrenamiento y validación en cada época, con los parámetros mencionados, como se puede observar en la imagen existe una curva descendente consistente en las perdidas hasta el 4to fold, donde Early Stopping detuvo el entrenamiento para así evitar el sobreajuste.

Una vez completado el entrenamiento en todos los pliegues, se guardó el mejor modelo encontrado para ser utilizado con los datos de prueba.

Redes neuronales: TabNet

TabNet es un modelo transformer diseñado específicamente para datos tabulares, introduce un mecanismo de atención secuencial que permite a la red enfocarse en las características más relevantes en cada paso, en lugar de procesar todas las características al mismo tiempo, a diferencia de NeuralNet.

Esta capacidad de atención secuencial permite a TabNet modelar de manera más efectiva las relaciones complejas entre las variables y capturar el orden significativo en las columnas de datos tabulares. En otras palabras, TabNet puede aprender a "leer" los datos tabulares de manera similar a como un humano leería una tabla, prestando atención a las partes más importantes y relevantes en cada momento.

Ventajas de TabNet:

- **Interpretabilidad:** TabNet proporciona información sobre la importancia de las características, lo que ayuda a comprender cómo el modelo toma decisiones.
- **Rendimiento:** En muchos casos, TabNet supera a otras arquitecturas de redes neuronales en tareas de clasificación y regresión con datos tabulares.
- **Eficiencia:** A pesar de su complejidad, TabNet puede ser entrenado de manera eficiente en conjuntos de datos grandes.

Para implementar TabNet se adaptó la arquitectura base de NeuralNet descrita anteriormente, para incorporar el mecanismo de atención secuencial de TabNet. Se mantuvieron constantes otros aspectos del entrenamiento y la configuración para hacer homologables los resultados de ambos modelos.

Evaluación de modelos

Para evaluar el desempeño de los modelos (regresión logística, árbol de decisión, NeuralNet y TabNet), se entrenaron y validaron utilizando tanto hardware local con GPU y CPU. Este enfoque híbrido permitió aprovechar la aceleración de la GPU para tareas computacionalmente intensivas, como el entrenamiento de las redes neuronales, y la flexibilidad de la CPU para otras etapas del proceso, como el entrenamiento de los modelos de Scikit-learn.

Una vez finalizado el entrenamiento, se procedió a una evaluación de los datos de entrenamiento y prueba. Se calcularon diversas métricas de rendimiento para cada modelo y dataset, incluyendo:

- Accuracy (Exactitud): Proporción de predicciones correctas sobre el total de predicciones.
- Precision (Precisión): Proporción de verdaderos positivos sobre el total de positivos predichos.
- Recall (Sensibilidad): Proporción de verdaderos positivos sobre el total de positivos reales.
- F1-score: Media armónica entre precisión y recall, útil para equilibrar ambas métricas.
- Curva ROC: Representación gráfica de la capacidad de discriminación del modelo a diferentes umbrales de clasificación.

las estrategias implementadas. Entre los principales logros, sobresale la consistencia observada entre los resultados obtenidos tanto en los conjuntos de entrenamiento como de prueba. Esto sugiere que las medidas aplicadas para evitar el sobreajuste, como la regularización y la selección adecuada de características, fueron exitosas. De este modo, los modelos lograron capturar patrones relevantes de los datos, manteniendo un equilibrio adecuado entre complejidad y generalización, sin caer en el subajuste. Estos resultados respaldan la capacidad de los modelos para predecir con precisión la asistencia o inasistencia a citas médicas, un componente crucial en la optimización de la asignación de recursos en los centros de salud.

Un aspecto fundamental que respaldó la validez de todos los modelos evaluados fue su rendimiento superior al azar, reflejado en valores de AUC (Área Bajo la Curva ROC) consistentemente por encima de 0.5. Este resultado confirmó su capacidad para discriminar entre las clases de interés de manera efectiva.

Sin embargo, el análisis del mapa de calor reveló un desafío persistente en términos de precisión. Esta métrica se mostró como la más difícil de optimizar para todos los modelos, independientemente del conjunto de datos utilizado. Incluso los esfuerzos específicos de optimización mediante búsqueda en cuadrícula (GridSearchCV) en los modelos de Scikit-Learn no lograron elevar la precisión por encima del 50%. Esta observación tuvo implicaciones prácticas significativas, ya que indicó que en más de la mitad de los casos en que se predijo la no asistencia de un paciente, este terminó asistiendo. Tal situación plantea retos importantes para la gestión eficiente de citas y recursos médicos.

En contraste con los desafíos en la precisión, el Recall emergió como la métrica más destacada en el rendimiento de los modelos. En particular, el modelo de NeuralNet alcanzó un valor de Recall del 83.7% en el conjunto de datos P1-ALL-ADASYN, que incluía todas las variables del dataset sin eliminaciones y en el que se aplicó ADASYN para el balanceo de clases. Se lograron valores notables, llegando hasta un 83.7% en la base de datos P1-ALL-ADASYN utilizando el modelo de NeuralNet. Este alto Recall fue particularmente valioso en el contexto del proyecto, ya que minimizó la predicción errónea de asistencias que no se materializaban, contribuyendo así a reducir los espacios vacíos en las agendas médicas y optimizó la utilización de recursos en la fase operativa del proyecto.

Entre los modelos evaluados, TabNet se distinguió por su rendimiento superior en métricas clave. Alcanzó el mejor AUC de 0.74 en el conjunto de datos P6A-KIDS, que correspondía a pacientes menores de 12 años, demostrando una excelente capacidad discriminativa. Además, TabNet logró los mejores F1 score, llegando hasta 0.49 en el conjunto de datos P6C-YOUNGADULTS, correspondiente a pacientes de entre 19 y 35 años, lo que indicó un mejor equilibrio entre Precisión y Recall en comparación con los otros modelos analizados. Esta consistencia en diferentes métricas posicionó a TabNet como el modelo más prometedor para cumplir con los objetivos del proyecto (objetivo Obj2.2).

Como discusión, quedó pendiente evaluar el rendimiento de modelos adicionales como Support Vector Machine (SVM), XGBoost y modelos ensamblados de Stacking. Estos enfoques no se valoraron hasta entonces debido a limitaciones de hardware, como se planteó anteriormente. Sin embargo, explorar estos modelos podría ofrecer nuevas perspectivas y mejoras adicionales en el rendimiento predictivo.

Análisis comparativo de resultados en otros modelos predictivos de No-show en citas médicas

En el artículo “Predicting No-show Medical Appointments Using Machine Learning” (Alshaya et al., 2019) (11), se desarrolló un modelo para predecir la inasistencia a citas médicas basado en un enfoque multi-etapa que incluyó reducción de dimensionalidad, balanceo de datos y diversas técnicas de clasificación. Este trabajo fue particularmente relevante para la comparación, ya que empleó la misma base de datos de Kaggle utilizada en nuestro proyecto y abordó desafíos similares relacionados con el desbalance de clases y la predicción de “NoShow”.

Modelos Comparados

Alshaya et al. evaluaron diversos algoritmos de machine learning, como regresión logística, Random Forest, Support Vector Machines (SVM), como también Naive Bayes, utilizando técnicas de balanceo como ADASYN, SMOTE, undersampling aleatorio y enfoques híbridos como SMOTE-ENN. En nuestro proyecto, TabNet fue el modelo más eficaz, alcanzando un AUC de 0.74, un valor comparable con los modelos de clasificación tradicionales evaluados en el estudio de Alshaya et al.

La siguiente tabla compara los resultados de diferentes modelos de ambos estudios:

Modelo	Técnica de Balanceo	AUC ROC	F1 Score
NeuralNet (TFM)	ADASYN	0,69	0,42
TabNet (TFM)	ADASYN	0.74	0,44
Random Forest (Alshaya et al., 2019)	SMOTEENN	0,65	0,36
Support vector classifier (Alshaya et al., 2019)	ADASYN, SMOTEENN, AllKNN, RandomUnderSampler	0.68	0,44
Stochastic gradient descent	SMOTE, SMOTEENN, SMOTETomek	0.68	0,44
Logistic Regression (Alshaya et al., 2019)	ADASYN, SMOTEENN, RandomUnderSampler	0,71	0,43
Naive Bayes (Alshaya et al., 2019)	ADASYN	0.86	-

Análisis de Técnicas de Balanceo

En nuestro proyecto, utilizamos SMOTE-ENN y ADASYN para manejar el desbalance de clases, mientras que Alshaya et al. exploraron métodos adicionales como el undersampling aleatorio y SMOTETomek. El uso de ADASYN en ambos estudios demostró proporcionar mejoras notables en el AUC, especialmente cuando se combinó con técnicas de reducción de dimensionalidad, como los Autoencoders (AE). Aunque no implementamos esta técnica en nuestro proyecto, su inclusión podría ser una mejora valiosa para futuras iteraciones, al permitir un balanceo más preciso de las clases y potencialmente aumentar el rendimiento general del modelo.

El uso de SMOTE-ENN en ambos estudios resultó ser una estrategia altamente efectiva, ya que combinó los beneficios de sobre muestreo y la eliminación de ruido en los datos. A pesar de que Naive Bayes logró AUC más alto en el artículo de Alshaya et al. (0.86), no se proporcionó un F1 Score, lo que limitó una evaluación completa en términos de precisión y recall.

Conclusiones

Comparado con los resultados reportados por Alshaya et al. (2019), nuestro enfoque con TabNet y NeuralNet mostró un rendimiento competitivo, especialmente en términos de AUC y F1 Score. Mientras que Naive Bayes alcanzó un AUC de 0.86, la falta de datos sobre el F1 Score sugiere que este modelo podía haber favorecido la clase mayoritaria. En cambio, TabNet logró un equilibrio más sólido entre precisión y recall gracias a las técnicas de balanceo implementadas, lo que lo convirtió en un modelo robusto para la predicción de no-shows. La combinación de técnicas como SMOTE-ENN y ADASYN demostró ser clave para obtener predicciones más precisas en la clase minoritaria, sugiriendo que estos enfoques, junto con modelos más modernos como TabNet, podían superar a los métodos tradicionales como la regresión logística en problemas de predicción de No-Show.

Fase 2. Implementación de un sistema de overbooking

Estudio de bibliografía relacionada

La extensa bibliografía desarrollada en relación con los Sistemas “System Appointment Scheduling” (SAS) es un buen indicador de la dificultad para resolver el problema, como también de la importancia que tiene mejorar el servicio en atención médica y disminuir costos asociados.

En “A Review of Optimization Studies for System Appointment Scheduling” ([12](#)) se definen las características principales de un SAS, destacando que, en los servicios ambulatorios, una cuestión central para la programación de las citas es cómo asignar los espacios de tiempo disponibles para los pacientes, reduciendo las demoras de los pacientes y la disponibilidad de los médicos o el tiempo adicional.

Son muchos los factores que complican la resolución real del problema, empezando por la selección del marco en el que se toman las decisiones para optimizar las citas, según el cual se identifican 3 tipos de decisiones:

- **Estratégicas:** decisiones a largo plazo que determinan la estructura principal del Sistema. Las políticas tradicionales implican que toda la capacidad de la clínica es cubierta con pacientes preasignados mediante cita previa. Existen otro tipo de centros médicos donde el acceso es abierto, y las consultas se van atendiendo a medida que los pacientes acceden al centro (walk-ins). También se pueden implementar estrategias híbridas, donde convivan pacientes con cita previa y pacientes que se presentan a consulta el mismo día (walk-ins), con más o menos restricciones al respecto. Nuestro proyecto de estudio se comporta como un sistema híbrido, ya que en la base de datos hay muchos pacientes que tienen el “AppointmentDay” el mismo día del “ScheduledDay”. Aceptar pacientes sin cita previa disminuye los costes de inactividad asociados al absentismo, pero aumenta la complejidad del modelo. Otra decisión estratégica dentro las políticas tradicionales es realizar una programación de citas “offline” o “online”. El método “offline” implica esperar a asignar las citas una vez se han recibido todas las solicitudes (sólo apto para usos muy particulares), mientras que en el método “online” todas las citas se asignan inmediatamente una vez se recibe la solicitud (nuestro caso).
- **Tácticas:** decisiones más a medio plazo para determinar cómo se utilizan los servicios, si se permite la elección de médico, si se otorga preferencia a determinados grupos (si se va a priorizar el acceso a consulta a pacientes con múltiples dolencias médicas, o a aquellos que tienen becas de acceso al centro médico, o a los pertenecientes a unas determinadas mutuas, etc.). Otra decisión táctica muy importante es definir si se permite o no overbooking, y en qué grado. En este caso particular, poner muchas restricciones al overbooking puede mejorar la calidad del servicio, pero seguramente a expensas de aumentar los costes de inactividad. La definición de la duración de los slots en los que se asigna y atiende a los pacientes, o de la diferenciación o no de los slots en bloques, también afecta considerablemente a la capacidad de la clínica.
- **Operativas:** son las decisiones a corto plazo que hay que tomar a la hora de asignar las citas. Si éstas se toman con enfoques basados en reglas bien definidas (RBA – Rule Based Approaches), o con enfoques más basados en la optimización de algún parámetro (OBA – Optimization Based Approaches). Las RBA son más sencillas de aplicar, pero no garantizan un rendimiento óptimo, mientras que las OBA están específicamente estudiadas para alcanzar el mejor comportamiento, aunque son mucho más difíciles de diseñar. Nuestro objetivo en este proyecto fue estudiar y comparar diferentes decisiones operativas, incluyendo RBA tradicionales y una OBA que incluyera la optimización de asignación de citas en función de predicciones de asistencia.

La siguiente tabla, extraída de “A Review of Optimization Studies for System Appointment Scheduling” (12), resume las principales características relacionadas con el marco de decisiones de un SAS.

Decision Types	Features
Strategic decisions	<ul style="list-style-type: none"> (a) Long-term decisions; (b) Determine the main system structure; (c) Include access policy; number of resources; policy on acceptance of walk-ins; types of scheduling.
Tactical decisions	<ul style="list-style-type: none"> (a) Medium-term decisions; (b) Explaining how patients are placed as a whole; (c) Maximizing resource utilization and the accessibility of nursing services.
Operational decisions	<ul style="list-style-type: none"> (a) Short-term decisions; (b) Focused on the efficient scheduling of individual patients; (c) Incorporating both the rule-based approach (RBA) and the optimization-based approach (OBA).

Figura 40. Marco de decisiones de un Sistema SAS

Así pues, la dificultad en la resolución del problema radica en la multitud de decisiones que se deben realizar para definir el problema de optimización estocástica, así como en el grado de incertidumbre que tienen los parámetros y variables que finalmente se decide tener en cuenta.

Parámetros que aportan mayor incertidumbre y variabilidad a los Sistemas SAS:

- **Incertidumbre en las predicciones de asistencia.** Los modelos de predicción están lejos de ser perfectos. Aunque los resultados son mejores que una mera asignación de probabilidad media de asistencia o una predicción al azar, las métricas de precisión y sensibilidad siguen proporcionando un grado de incertidumbre bastante elevado, y que será automáticamente trasladado al modelo de optimización.
- **Variedad e incertidumbre en los factores ambientales.** El número de factores ambientales que se pueden aplicar al sistema son muy numerosos, y seguramente diferentes para cada centro médico o de salud, pues dependen de las decisiones estratégicas y tácticas mencionadas más arriba. Adicional, muchos de ellos añaden unos valores de incertidumbre que ni siquiera están estudiados por modelos predictivos. Así pues, podemos distinguir entre factores más certeros (número de servicios integrados, número de doctores, número de citas por sesión, prioridad en la atención) y más inciertos (puntualidades, tiempos de atención, nivel de interrupción de los doctores, llegadas espontáneas).
- **Variedad en las reglas de cita y en reglas de secuencia.** Las reglas de cita definen las restricciones que imponemos al modelo para limitar las asignaciones (número de pacientes que se puede asignar a un único slot, duración de los slots, etc.), y las reglas de secuencia definen el orden con los que los pacientes son asignados a los slots en función de una determinada clasificación (prioritarios, primerizos, recurrentes, etc.). Forman parte de las decisiones operativas mencionadas por Tiantian et al (12). Sólo Cayirli, Veral y Rosen (13) ya realizaron un estudio probando 6 diferentes reglas de secuencia con 7 reglas de cita, lo que hace un total de 42 Sistemas de Asignación (AS, del inglés) diferentes. Para hacernos otra idea de la magnitud y complejidad del problema, Ho y Lau (14) comparaban 9 diferentes sistemas de citas en 27 diferentes escenarios clínicos caracterizados por 3 factores ambientales (probabilidad de ausencias, variación de tiempos de atención y el número de pacientes por sesión).
- **Definición del Coste.** Para minimizar la Función de Coste que permite realizar la mejor programación de citas médicas se tiene que definir primero cuál es dicho coste y cómo calcularlo, lo cual no es banal. Y como se indica a continuación, las posibilidades son prácticamente infinitas.

Para empezar, según Tiantian et al. (12) la optimización varía en función del objetivo que se pretenda conseguir: beneficio social, rentabilidad económica de la clínica, máxima amortización de recursos, etc. Así pues, Chew (15) y Kaandorp and Koole (16) dividen la función en 3 tipologías de coste distintas: el tiempo de espera del paciente en consulta, el tiempo de inactividad de toda la infraestructura médica por falta de paciente, y el tiempo extra dedicado por la infraestructura médica a atender los pacientes después de una jornada laboral normal. Ho y Lau (14) realizan de forma similar, pero sin tener en cuenta el tiempo extra de los médicos. Y Harris and Samorani (17) utilizan los mismos conceptos, pero sin tener en cuenta el tiempo de inactividad de la infraestructura médica. Con otra perspectiva, Almaktoom (18) calcula el coste respecto a los costos de realizar una consulta, los de operación de la clínica y los de overbooking, referenciándolos al número de No-Shows y de Overbookings. Sin embargo, en Valenzuela-Núñez et al. (19) no consideran minimizar un coste, sino maximizar la utilidad que genera el centro sanitario por cada cita atendida, menos una penalidad por sobrecarga de overbooking, directamente proporcional a los casos de overbooking registrados. Y Lawley and Muthuraman (20) usaron un modelo estocástico para minimizar el número de pacientes desbordados de una cita a la siguiente.

Elección de las hipótesis para la resolución del problema y fórmula para el cálculo del Coste del Sistema a minimizar

Para definir el Problema de Optimización Estocástico se tiene que decidir sobre:

- **Función objetivo:** lo que se quiere optimizar, relacionado con los marcos de optimización mencionados por Tiantian et al. (12).
- **Variables de decisión:** aquello que se pretende ajustar en el modelo. En nuestro caso, la distribución de pacientes en los slots que dura una consulta médica.
- **Restricciones:** limitaciones con las que se restringen las posibles soluciones. Estas restricciones pueden ser el número de pacientes esperando, la capacidad máxima de pacientes que se puede atender en un día, o el horario máximo de atención, por ejemplo. Y formarán parte de las hipótesis a fijar en este apartado.
- **Variables aleatorias:** los elementos que están sujetos a variabilidad y que se modulan mediante distribuciones de probabilidad.

En nuestro caso, en aras de simplificar la resolución del problema, se minimizaron las variables del modelo como sigue:

Factores ambientales

- Predicciones de Asistencia. Extraídas de la Fase 1.
- Tiempo de Atención al Paciente = 20 min. Igual para todos los pacientes sin excepción.
- Horario de Atención normal. De 9:00 a 13:00 y de 15:00 a 19:00 (8 horas, separadas en 2 bloques de 4 horas).
- Número de slots por sesión = 24 (consideradas las consultas de veinte minutos en una jornada normal de 8 horas).
- Número de slots por bloque = 12 (dos bloques de 4 horas por sesión, consideradas las consultas de veinte minutos).
- El resto de los factores los anulamos directamente:
 - No hay impuntualidad.
 - Tiempo de consulta fijo e invariable para todo paciente. No hay demoras en la atención, ni las consultas terminan antes de tiempo.
 - Un único servicio integrado.

Reglas de Asignación de Citas

Se definieron 3 escenarios donde aplicar distintas reglas de asignación de citas. Cada escenario, a su vez, contenía diferentes casos de uso (salvo el primero, que es de caso único). Estas reglas de asignación son las que nos permitieron configurar las agendas médicas, sobre las cuales se calcularon, a posteriori, los Costes del Sistema generado, en función de las asistencias reales de los pacientes.

Para poder hacer un mejor seguimiento de las Agendas Médicas creadas, y sus costes asociados, se definió una nomenclatura para cada caso dentro de los 3 escenarios. Para los escenarios 1 y 2, basados en RBA tradicionales, fue como sigue:

- Una “R” de regla.
- Una serie de pares de dígitos. Dentro de cada par el ‘primer dígito’ es cuantos slots seguidos van a tener x números de pacientes asignados (‘segundo dígito’). Cuando el número de pacientes asignados a un slot cambia, se pasa a indicar el siguiente par de dígitos, a no ser que se pueda completar la serie de media jornada (12 slots) repitiendo los pares ya indicados, o añadiendo un único paciente a cada slot faltante.
- Un guion bajo “_” y el número total de pacientes asignados a la consulta médica (un día con 24 slots de 20 min cada uno).

Pero en este caso vale más una imagen que mil palabras:

Regla Asignación Cita	Número Slot - Bloque Mañana (9:00 a 13:00)	Descanso	Número Slot - Bloque Tarde (15:00 a 19:00)
	1 2 3 4 5 6 7 8 9 10 11 12 Comida	13 14 15 16 17 18 19 20 21 22 23 24
Escenario 1 (RBA Tradicional sin Overbooking):			
R11_24	1 1 1 1 1 1 1 1 1 1 1 1		1 1 1 1 1 1 1 1 1 1 1 1 = 28 pacientes
Escenario 2 (RBA Tradicional con Overbooking):			
R1231_28	2 1 1 1 2 1 1 1 1 1 1 1		2 1 1 1 2 1 1 1 1 1 1 1 = 28 pacientes
R1241_28	2 1 1 1 1 2 1 1 1 1 1 1		2 1 1 1 1 2 1 1 1 1 1 1 = 28 pacientes
R1221_30	2 1 1 2 1 1 2 1 1 1 1 1		2 1 1 2 1 1 2 1 1 1 1 1 = 30 pacientes
R1231_30	2 1 1 1 2 1 1 1 2 1 1 1		2 1 1 1 2 1 1 1 2 1 1 1 = 30 pacientes
R133112311231_32	3 1 1 1 2 1 1 1 1 2 1 1		3 1 1 1 2 1 1 1 1 2 1 1 = 32 pacientes
R133112211241_32	3 1 1 1 2 1 1 2 1 1 1 1		3 1 1 1 2 1 1 2 1 1 1 1 = 32 pacientes

Figura 41. Reglas Asignación de Citas para los Escenarios 1 y 2.

Para el escenario 3 la nomenclatura fue algo más simple, aunque da pie a un número de casos mucho mayor:

- “**Prob_Show**”, para indicar que es una OBA basada en la probabilidad de asistencia (Show) de cada paciente.
- Probabilidades que definen la función lineal o cuadrática del “listón de probabilidades” bajo el cual se regirá la asignación de pacientes (se entra más en detalle de su significado en la explicación del escenario 3, dentro de este apartado).
 - Si son dos números separados por un guión bajo “_” significa que el “listón de probabilidades” está definido mediante una función lineal. El primer número es el listón asignado a los slots 1 y 13, mientras que el segundo número es el listón asignado a los slots 12 y 24.
 - Si son tres números, separados igualmente por un guión bajo “_”, el “listón de probabilidades” está definido mediante una función cuadrática. El primer número es el listón asignado a los slots 1 y 13, el tercer número es el listón asignado a los slots 12 y 24, mientras que el segundo número es el listón asignado en el punto medio de ambos rangos.

La siguiente Figura muestra un ejemplo para los casos “**ProbShow 0,8_0,15**” (función lineal) y “**ProbShow 0,85_0,25_0,15**” (función cuadrática):

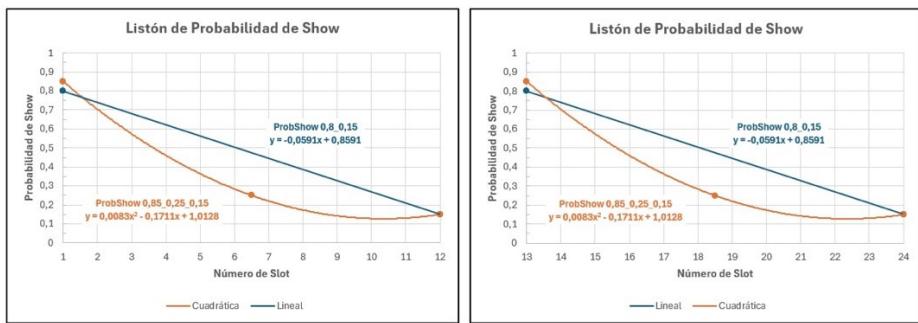


Figura 42. Ejemplo de caso de Escenario 3 con Probabilidad de Show.

Escenario 1 – RBA Tradicional sin Overbooking (caso R11_24)

Sólo se permite un paciente por slot, sin posibilidad de espera por parte de los pacientes ni tiempos de atención extra, pero sí mucho coste por inactividad de los servicios médicos. Esto equivale a una asignación de 24 pacientes (24 slots) por servicio médico y día.

Escenario 2 – RBA Tradicional con Overbooking (casos R1231_28, R1241_28, R1221_30, R1231_30, R1331123112_32 y R1331122112_32)

Sabiendo que hay pacientes que no asisten a su cita médica, se contempló la posibilidad de generar overbooking en algunos slots. Como la probabilidad promedio de No Show en nuestro dataset es del 20,19%, se decidió iniciar programando más pacientes a la consulta médica a la razón de 1,2019 pacientes extra, lo que equivale a una asignación de 30 pacientes por servicio médico y día, lo que, al estar distribuidos entre 24 slots, genera el overbooking. La regla de asignación mencionada es la de 1 slot con 2 pacientes, 3 slots con un paciente, y así sucesivamente hasta completar la asignación de los 30 pacientes, pero de igual forma se generaron agendas médicas con otras reglas con menos y más pacientes convocados. La Figura 41 muestra las diferentes Reglas de Asignación de Citas contempladas en este estudio.

Escenario 3 – OBA con Probabilidades de Asistencia con Overbooking (casos ProbShow) (Obj3.1)

En este caso no se hace una asignación siguiendo una regla de orden natural, sino que incluimos la variable de probabilidad de asistencia a la cita, determinada mediante el Modelo de ML de la Fase 1, para determinar si un paciente tiene cabida o no en un determinado slot. A la hora de asignar un paciente a una agenda médica, se suma, en los slots ya abiertos, las Probabilidades de Show de los pacientes ya asignados al slot en estudio y la del propio paciente que se está asignando, incluyéndolo en dicho slot sólo cuando dicho valor no supera un cierto valor del “Listón ProbShow”.

Este “Listón ProbShow” no es igual para cada slot, pues para maximizar la utilización de los recursos del centro médico y minimizar el coste interesó asignar más pacientes en los primeros slots (para minimizar costes de inactividad) y menos en los últimos (para minimizar costes de tiempo extra), por lo que lo definimos como una función (lineal o cuadrática, según el caso estudiado) que va de más a menos, diferenciándola para los 2 bloques de slots definidos (del 1 al 12, y del 13 al 24). La Figura 42 muestra un par de ejemplos de los listones ProbShow estudiados.

Si después de repasar todos los slots abiertos de una consulta, se determinaba que el paciente no cabía en ninguno de ellos, se le asignó al siguiente slot disponible (siempre y cuando no se hubieran abierto ya los 24 slots de una consulta).

Adicional, para evitar controlar mejor los costes de espera de los pacientes, (y a su vez evitar situaciones incómodas con los mismos), se incluyó una Restricción de Contorno extra en la que no se permitió agendar más de 3 pacientes para un mismo slot.

Reglas de secuencia

Utilizamos una secuencia FCFA (First Call First Appointment), también conocida directamente como Regla de No Secuencia, pues simplemente se asignó al paciente en el mejor slot disponible calculado mediante el modelo entrenado de overbooking (escenario 3), o según la Regla de Asignación de Cita (RBA) definida en el punto anterior para los escenarios 1 y 2.

Función de coste (Obj3.2)

Utilizaremos una fórmula parecida a la usada por Chew (15), dividiendo y calculando la función de coste en 3 términos independientes:

- **T_w = Tiempo de espera del paciente (Wait Cost).** Calculado como unidad temporal que debe esperar un paciente a ser atendido por encontrarse el slot sobrecargado. Se entiende que la paciencia de los pacientes no es indefinida, y, por lo tanto, el valor que tiene su tiempo no es igual si tiene que esperar una sola vez, a que tenga que esperar múltiples slots para ser atendido. Es por ello por lo que se hizo una elevación cuadrática a la unidad temporal que tiene que esperar, siendo igual a 1 si sólo tiene que esperar 1 slot, 3 si tiene que esperar 2 slots, 6 si tiene que esperar 3 slots, etc. Dicha función queda representada por la siguiente fórmula:

$$T_{w_i}(y) = \frac{y^2 + y}{2}$$

y = número de slots a esperar por un paciente i

- **T_o = Tiempo extra a realizar por el doctor (Overtime Cost).** Calculado como unidad temporal que tiene que trabajar fuera de la sesión o jornada laboral normal de 8 horas. Si no tiene que trabajar ningún slot será igual a cero, si tiene que trabajar un slot será igual a 1, y así sucesivamente. (unidad temporal = 20 min = 1 slot).
- **T_I = Tiempo de inactividad en la clínica (Idle Cost).** Calculado como unidad temporal sin estar atendiendo a ningún paciente. Si el doctor está siempre ocupado será igual a cero, si queda un slot vacío será igual a 1, y así sucesivamente. (unidad temporal = 20 min = 1 slot).

A cada término independiente le dimos unos costes monetarios distintos, pues no todos los tiempos tienen el mismo valor. Se entiende de que hay valores muy subjetivos en el cálculo de dichos importes, como pueden ser el coste que tiene para un paciente la hora de espera, la hora extra de trabajo para un médico, o lo que cuesta el “desperdicio” de recursos médicos en la Sociedad, pero consideramos que la siguiente fue una buena aproximación:

- **$Ic / minuto = 16,00 €/min$,** considerando el costo de un médico sin pasar consulta durante 20 min en unos 100 € (5 €/min), más 1 €/min de otros costes operativos de la clínica, más 10 €/min extras de coste de oportunidad, lo que centra el foco de atención en maximizar el uso de los recursos de los centros médicos a disposición de los pacientes (**objetivo Obj3.3**).
- **$Wc / minuto = 0,50 €/min$,** considerando que el componente cuadrático de este coste ya está contemplado en el cálculo de T_w .
- **$Oc / minuto = 3,50 €/min$,** considerando como sobrecosto de “Overtime” únicamente la parte proporcional extra que se paga al doctor, es decir, la mitad de los 5 €/min (considerando que se le paga su sueldo x1,5), más el mismo 1 €/min extra de otros costes operativos del centro médico.

$$\begin{aligned}Coste Total &= T_I(x) * 16 \text{ €/min} * 20\text{min} + \\&+ \sum_{i=1}^N T_{w_i}(y) * 0,5 \text{ €/min} * 20\text{min} + \\&+ T_O(z) * 3,5 \text{ €/min} * 20\text{min}\end{aligned}$$

$$\begin{aligned}
 T_i(x) &= x = \text{número de slots de inactividad.} \\
 y &= \text{número de slots a esperar por un paciente } i. \\
 N &= \text{número total de pacientes en una consulta.} \\
 T_0(z) &= z = \text{número de slots a realizar tiempo extra.}
 \end{aligned}$$

Cálculo del Coste del Sistema en Métodos Tradicionales (RBA) de asignación de citas médicas

Los métodos tradicionales de asignación de citas médicas corresponden a los Escenarios 1 y 2 citados anteriormente. Para calcular el coste se necesita realizar una serie de tareas previas, explicadas a continuación.

Revisión del Set de Pruebas para Fase 2

Primero se cargó el Set de Pruebas heredado de la “Fase 1. Predicción de Asistencia a Citas Médicas”, seleccionando sólo las variables que necesitábamos para calcular los Costes del Sistema SAS: 'PatientId', 'ScheduledDay', 'AppointmentDay', 'NoShow'.

A continuación, se agruparon los datos por “AppointmentDay”, y se revisó el número de citas y los porcentajes de “NoShow” según fecha de cita. De esta forma se observó que para los 2 primeros días de cita hay muy pocas citas en la base de datos, y prácticamente todas son No Show. Esto se debe a la forma en que se generaron los Sets de Entrenamiento y Prueba en la Fase 1. Cabe recordar que la partición se hizo para meter en el Set de Pruebas las últimas citas de pacientes únicos, en estricto orden descendente de la fecha de cita, y, al forzar para que tuviese exactamente el mismo porcentaje de “NoShow”s que el dataset original, se fue completando con pacientes “NoShow” de fechas más antiguas. Debido a esta situación se decidió descartar del Set de Pruebas las citas correspondientes a dichos días (ver la siguiente Figura).

```

Sets de Prueba completos:
El set para el 2016-05-30 tiene 60 citas, con un porcentaje de NoShows del 100.00%.
Descartamos los datos del 2016-05-30 por insuficiencia de datos y/o por haber excesivos No Show.
El set para el 2016-05-31 tiene 765 citas, con un porcentaje de NoShows del 66.80%.
Descartamos los datos del 2016-05-31 por insuficiencia de datos y/o por haber excesivos No Show.
El set para el 2016-06-01 tiene 3061 citas, con un porcentaje de NoShows del 18.75%.
El set para el 2016-06-02 tiene 3248 citas, con un porcentaje de NoShows del 18.29%.
El set para el 2016-06-03 tiene 3163 citas, con un porcentaje de NoShows del 19.60%.
El set para el 2016-06-06 tiene 3802 citas, con un porcentaje de NoShows del 18.73%.
El set para el 2016-06-07 tiene 3884 citas, con un porcentaje de NoShows del 17.64%.
El set para el 2016-06-08 tiene 4114 citas, con un porcentaje de NoShows del 17.14%.

```

Figura 43. Fechas descartadas del Set de Pruebas

Esta actuación rebajó el porcentaje de “NoShows” del 20,18% en el Set de Pruebas original al 18,29% en el nuevo conjunto, pero esto carece de relevancia para el cálculo y comparativa de costes en los diferentes escenarios.

Formación de Agendas Médicas en función de las Reglas de Asignación

La asignación de pacientes a los diferentes slots de una consulta médica se realizó en función de las Reglas de Asignación de Citas definidas para cada Escenario, aplicando la Regla de Secuencia FCFA (First Call First Appointment).

Para aplicar la Regla de Secuencia FCFA primero se ordenó el Set de Pruebas por fecha de solicitud de cita (“ScheduledDay”), y así poder iterar paciente por paciente en estricto orden de llamada.

Para el Escenario 1 – Tradicional sin Overbooking, sólo existe una regla de asignación (R11_24), ya que simplemente se trata de ir asignando a cada paciente el primer slot vacío de una consulta. En caso de no quedar slots disponibles, se consideró como hipótesis que tenemos más doctores disponibles y, por lo tanto, se abre otra consulta. Así indefinidamente hasta que se asignaron todos los pacientes a una consulta para la fecha solicitada (“AppointmentDay”).

En el Escenario 2 – Tradicional con Overbooking, se realizó exactamente la misma operación, sólo que, para algunos slots (en función de la Regla de Asignación de Citas), en lugar de asignar un solo paciente, se asignó una lista de 2 o 3 pacientes, según el caso.

Las Agendas Médicas generadas fueron diccionarios con la siguiente estructura:

```
Agenda_medica_R1231_30 = {
    "2016-06-01": {
        "consulta_001": {
            "slot_01": [id_paciente, id_paciente],
            "slot_02": [id_paciente],
            ...
            "slot_24": [id_paciente]
        },
        ...
        "consulta_032": {
            "slot_01": [id_paciente, id_paciente],
            ...
        }
    },
    ...
    "2016-06-08": {
        "consulta_001": {
            "slot_01": [id_paciente],
            ...
        }
    }
}
```

Aunque posteriormente estos diccionarios se aplanan para poder generar bases de datos tabulares y guardar las agendas en archivos tipo Excel.

Es relevante señalar que el conjunto de datos original carece de información sobre la hora de las citas médicas, limitándose a la fecha. En el caso de que los pacientes deseen agendar la cita médica a una hora particular, lo cual es habitual, no será posible aplicar las Reglas de Asignación. No obstante, se puede seguir utilizando el Escenario 3, ya que este enfoque no depende de una regla de asignación de citas (RBA), sino de un método optimizado (OBA) basado en la probabilidad de asistencia a un slot específico y en el “listón de probabilidad” previamente definido. Esta consideración es fundamental para la futura aplicación real del estudio.

Cálculo de Costes

Finalmente se aplicó la función de coste a cada una de las agendas generadas para los Escenarios 1 y 2. En el anexo correspondiente al cuaderno Jupyter Notebook “[Schedule_Optimization_RBA.ipynb](#)” se puede consultar el código aplicado para calcular dicho coste.

El siguiente resumen sintetiza el proceso:

- Se itera para cada consulta de cada fecha slot por slot, y paciente por paciente asignado a cada slot.
- Se comprueba si el slot está ocupado en consulta médica y si el paciente asistió a la cita.
- Si el slot está libre y el paciente asiste, se disminuye el **Tiempo de Inactividad**.
- Si el slot está ocupado y el paciente asiste, se traslada este paciente a la cabeza de fila del siguiente slot, y se empieza a calcular el **Tiempo de Espera**, aplicando la fórmula exponencial en función del número de slots que le va a tocar esperar.
- Si el paciente no asiste, se pasa al siguiente paciente del slot.
- Cuando se acaban los pacientes de un slot, se pasa al siguiente slot, teniendo en cuenta los cambios habidos por transferencia de pacientes.

- Para los slots 12 y 24 se calcula también el **Tiempo Extra**, identificando el número de pacientes que quedan por ser atendidos en dichos slots.
- En el Slot 12 añadimos una restricción sobre la cual no se pueden atender a más de 3 pacientes después de las 13:00 horas. El resto de los pacientes en espera, de existir, se mueven directamente al Slot 13, para ser atendidos a partir de las 15:00.

En estos 2 escenarios se calcularon los costes de todas las consultas médicas que se generan con el Set de Pruebas disponible (descartando sólo la última porque no queda completa):

- Escenario 1: 147 consultas x 6 días contemplados = 882 consultas médicas.
- Escenario 2:
 - Casos con 28 pacientes: 126 consultas x 6 días contemplados = 756 consultas médicas.
 - Casos con 30 pacientes: 118 consultas x 6 días contemplados = 708 consultas médicas.
 - Casos con 32 pacientes: 110 consultas x 6 días contemplados = 660 consultas médicas.

Los resultados para el Escenario 1 y del Escenario 2 se muestran en las siguientes figuras.

COSTES EN CITAS MÉDICAS POR NO ASISTENCIA Y OVERBOOKING											
REGLA DE ASIGNACIÓN	Media consultas médicas por día	Media nº slots afectados para sobrecoleste		C _i : Coste Inactividad (consulta médica)		C _{wi} : Coste de Espera (pacientes)		C _t : Coste Tiempo Extra (consulta médica)		COSTE TOTAL	
		Inactividad	Tiempo Extra	Media	Desv. Est.	Media	Desv. Est.	Media	Desv. Est.	Media	Desv. Est.
Regla R11_24 pacientes	147	3,83	0,00	1.224,20	101,18	0,00	0,00	0,00	0,00	1.224,20	101,18
Coste Idle Cost / minuto = 16,00 €											
Coste una consulta = 5 € x 20 min = 100 € + otros Costes Operativos Fijos (1 €/min) + coste Oportunidad (10 €/min)											
Coste Waiting Cost / minuto = 0,50 €											
Coste con incremento exponencial: $(x2 + x) / 2$, siendo x el número de slots de espera.											
Coste Overtime / minuto = 3,50 €											
Coste extra pagado al doctor = (5 € x 0,5) x 20 min = 50 € + otros Costes extras por Uso Instalaciones (1 €/min)											
Idle Costs / minuto:											
5,00 € Coste Médico											
1,00 € Costes Operativos Fijos											
10,00 € Coste Oportunidad (pacientes no atendidos)											
Overtime Costs / minuto:											
2,50 € Coste extra Médico											
1,00 € Coste extra por uso de Instalaciones (Personal, Consumos generales)											

Figura 44. Coste Escenario 1 - Tradicional RBA sin Overbooking.

COSTES EN CITAS MÉDICAS POR NO ASISTENCIA Y OVERBOOKING											
REGLA DE ASIGNACIÓN	Media consultas médicas por día	Media nº slots afectados para sobrecoleste		C _i : Coste Inactividad (consulta médica)		C _{wi} : Coste de Espera (pacientes)		C _t : Coste Tiempo Extra (consulta médica)		COSTE TOTAL	
		Inactividad	Tiempo Extra	Media	Desv. Est.	Media	Desv. Est.	Media	Desv. Est.	Media	Desv. Est.
Regla R1231_28 pacientes	126	2,17	1,18	694,29	50,84	204,52	28,64	82,90	12,12	981,72	84,00
Regla R1241_29 pacientes	126	2,19	1,21	701,69	51,13	189,45	26,20	84,52	12,30	975,67	82,70
Regla R1221_30 pacientes	118	1,60	2,05	512,23	42,52	360,06	52,19	143,71	21,19	1.016,01	102,30
Regla R1231_30 pacientes	118	1,69	2,14	539,49	42,33	295,97	41,31	149,68	21,43	985,14	92,60
Regla R133112311231_32 pacientes	110	1,15	3,02	367,98	29,17	671,90	95,81	211,22	28,66	1.251,10	127,95
Regla R133112211241_32 pacientes	110	1,11	2,98	356,24	27,20	706,88	100,57	208,66	28,60	1.271,77	131,46
Coste Idle Cost / minuto = 16,00 €											
Coste una consulta = 5 € x 20 min = 100 € + otros Costes Operativos Fijos (1 €/min) + coste Oportunidad (10 €/min)											
Coste Waiting Cost / minuto = 0,50 €											
Coste con incremento exponencial: $(x2 + x) / 2$, siendo x el número de slots de espera.											
Coste Overtime / minuto = 3,50 €											
Coste extra pagado al doctor = (5 € x 0,5) x 20 min = 50 € + otros Costes extras por Uso Instalaciones (1 €/min)											
Idle Costs / minuto:											
5,00 € Coste Médico											
1,00 € Costes Operativos Fijos											
10,00 € Coste Oportunidad (pacientes no atendidos)											
Overtime Costs / minuto:											
2,50 € Coste extra Médico											
1,00 € Coste extra por uso de Instalaciones (Personal, Consumos generales)											

Figura 45. Costes Escenario 2 - Tradicional RBA con Overbooking.

Cálculo del Coste del Sistema con función basada en las predicciones de asistencia (Fase 1) para la asignación de citas médicas (OBA)

Este apartado se corresponde al Escenario 3 – Usando Probabilidades de Asistencia con Overbooking (casos ProbShow), donde se aprovechan los cálculos de predicciones de asistencia obtenidas en la Fase 1 del Proyecto para optimizar las asignaciones de cita de los pacientes.

Revisión del set de Pruebas para Fase 2

Para poder comprobar los resultados a posteriori, en el Escenario 3 se usó el mismo Set de Pruebas utilizado en los Escenarios 1 y 2, pero completado con la columna correspondiente a la probabilidad de NoShow calculada con el mejor modelo de IA obtenido en la Fase 1, pues es la base del nuevo método para la asignación de pacientes en las agendas médicas.

Una vez completado el Set de Pruebas con esta nueva variable “Prob_NoShow”, se siguió exactamente el mismo procedimiento. Así pues, los Sets de Validación de los diferentes escenarios son iguales, aunque en este escenario se haya añadido la variable “Prob_NoShow”.

Formación de Agendas Médicas en función de las Probabilidades de Asistencia de los Pacientes

Al igual que para los métodos tradicionales con Reglas de Asignación de Citas, primero se ordenó el Set de Pruebas por fecha de solicitud de cita (“ScheduledDay”), y así poder iterar paciente por paciente en estricto orden de llamada.

Adicional, se calculó la Probabilidad de Show como [ProbShow = 1 – Prob_NoShow], que es la que se utiliza para medir cuando un grupo de pacientes excede o no el “Listón de Probabilidades” (también llamado “Listón ProbShow”).

Por otro lado, se definió la función del “Listón ProbShow” como una Interpolación Polinómica en la forma de Lagrange (ver [Figura 46](#)), según la cual, para “n” puntos dados, se obtiene una interpolación polinómica de grado “n-1” que pasa por todos dichos puntos. Así pues, si se fija el listón de probabilidad de asistencia máximo para los slots 1 y 12, obtenemos la línea recta (función polinómica de grado 1) que define el listón de probabilidad de asistencia máximo para todos los slots intermedios. Pero si al mismo tiempo fijamos el listón de probabilidad de asistencia en un tercer punto intermedio, se obtiene en su lugar una función cuadrática (función polinómica de grado 2) para el “Listón ProbShow”.

```
def calculo_liston_ProbShow(x_values: list, y_values: list, x: int) -> float:
    """
    Calcula la coordenada "y" de una función polinómica para una coordenada "x" dada,
    usando cualquier número de puntos (Interpolación polinómica de Lagrange)

    Parámetros:
        x_values: Lista de coordenadas "x" de los puntos conocidos por donde debe pasar
            la función polinómica.
        y_values: Lista de coordenadas "y" de los 3 puntos conocidos por donde debe pasar
            la función polinómica.
        x: La coordenada para la cual se desea calcular la coordenada "y" de la
            función polinómica de Lagrange.

    Retorna:
        float: La coordenada "y" correspondiente en la función polinómica definida
        por un número indeterminado de puntos.
    ...
    grado_funcion = len(x_values)
    y = 0.0

    for i in range(grado_funcion):
        # Calcula el polinomio l_i(x)
        l_i = 1
        for j in range(grado_funcion):
            if i != j:
                l_i *= (x - x_values[j]) / (x_values[i] - x_values[j])
        # Añade el término correspondiente a y_i
        y += y_values[i] * l_i

    return y
```

[Figura 46. Interpolación Polinómica en la forma de Lagrange](#)

En el Escenario 3 se generaron numerosos casos con diferentes listones, tanto lineales como cuadráticos, con el fin de buscar y encontrar el mínimo Coste del Sistema SAS.

En este estudio se hizo manualmente caso a caso, pero se puede aplicar un algoritmo matemático (tipo descenso del gradiente aplicado en ML) que vaya iterando con los valores del “Listón ProbShow” en dos slots (para la función lineal), o en tres slots (para la función cuadrática), para encontrar los valores que minimicen la Función de Coste. Para este estudio no se tuvo la potencia informática necesaria para realizarlo, pues se requiere mucha capacidad de memoria para realizar todos los cálculos de una vez. Hay que tener en cuenta que en este escenario cada caso para calcular los costes requiere primero realizar la agenda médica (con una media de 50 consultas por día equivale a realizar la agenda médica de 300 consultas), más luego calcular su coste medio por consulta, tras lo cual seguir con la iteración/optimización mediante el Descenso del Gradiente y calcular un caso nuevo. Con nuestro equipo de cómputo no fuimos capaces de realizar más de 5 agendas médicas de casos distintos al mismo tiempo sin desbordar el equipo, y eso sin el proceso de cálculo de costes ni el cálculo del Descenso del Gradiente.

Otro paso para optimizar el coste sería iterar con funciones polinómicas de grado superior, pero el número de variantes a desarrollar es demasiado alto como para ser capaces de, mediante iteraciones manuales, encontrar el mínimo global de la función. La función polinómica ideal sería una de grado 11, que es la que nos proporcionaría el grado de libertad máximo para cada uno de los 12 slots que definen un bloque, y diferenciar la función del bloque 1 de la del bloque 2 con valores distintos, ya que en el Escenario 3 se usa la misma función para ambos bloques (Ver [Figura 42](#)).

Las Agendas Médicas generadas en este Escenario 3 fueron diccionarios que difieren un poco a los de los Escenarios 1 y 2, ya que para cada slot no sólo se guardaron los id's de los pacientes, sino también la probabilidad de asistencia a la consulta de dicho paciente, pues es un dato crucial para poder seguir asignando pacientes. Concretamente presentan la siguiente estructura:

```
Agenda_medica_ProbShow_0.85_0.25_0.15 = {
    "2016-06-01": {
        "consulta_001": {
            "slot_01": [(id_paciente, ProbShow), (id_paciente, ProbShow)],
            "slot_02": [(id_paciente, ProbShow), (id_paciente, ProbShow)],
            "slot_03": [(id_paciente, ProbShow)],
            ...
            "slot_24": [(id_paciente, ProbShow)]
        },
        "consulta_002": {
            "slot_01": [(id_paciente, ProbShow), (id_paciente, ProbShow), (id_paciente, ProbShow)],
            ...
        },
        ...
        "consulta_034": {
            "slot_01": [(id_paciente, ProbShow)],
            ...
        }
    },
    ...
    "2016-06-08": {
        "consulta_001": {
            "slot_01": [id_paciente, ProbShow]),
            ...
        }
    }
}
```

Al igual que en los Escenarios 1 y 2, estos diccionarios se aplanaron posteriormente para poder generar bases de datos tabulares y guardar las agendas en archivos tipo Excel.

Cálculo de Costes

El procedimiento para el cálculo de los costes fue igual al utilizado en los Escenarios 1 y 2, solo que en este escenario no se aplicó a todas las consultas posibles de las Agendas Médicas. Analizándolas para cada sub-escenario de ProbShow, se observó que las primeras consultas generadas son las que tenían más pacientes asignados. Esto se debe a que, cuando un paciente solicita una consulta para un día determinado, lo primero que se hace es ir repasando los slots ya generados, uno a uno, desde la primera consulta, para ver si encaja en alguno de ellos. Por lo tanto, es natural que las primeras consultas fueran las más saturadas, dejando siempre a los pacientes con mayores probabilidades de asistencia al final del recorrido (pues ya no caben en los slots abiertos). En la realidad esto no sucede así, ya que no se pueden abrir consultas médicas de forma indefinida (existe un número limitado de doctores, siendo incluso sólo uno cuando se busca una atención especializada), y lo que se hace realmente es buscar otro día de consulta. La realidad se comporta más como las primeras consultas de las series, pues siempre se intenta copiar la agenda médica de un día (u hora) determinados.

Por lo tanto, para evitar esta distorsión de la realidad, se descartó el último 50% de consultas generadas, así como el primer 10% de las mismas, ya que tampoco se prevé tener un número tan grande de solitudes para un mismo día y doctor como para ir incrustando los pacientes al límite de su “Listón ProbShow”, y mucho menos si finalmente se aplica la optimización por slot de consulta. Esa zona es donde, de media, se conserva una distribución del número de pacientes más uniforme y parecida a la realidad, tal y como se aprecia en las siguientes Figuras. Las barreras se definieron para posicionarse automáticamente de forma particular para cada agenda y para cada fecha, maximizando el número de consultas sobre las que se calcula el coste, robusteciendo su media y desviación estándar.

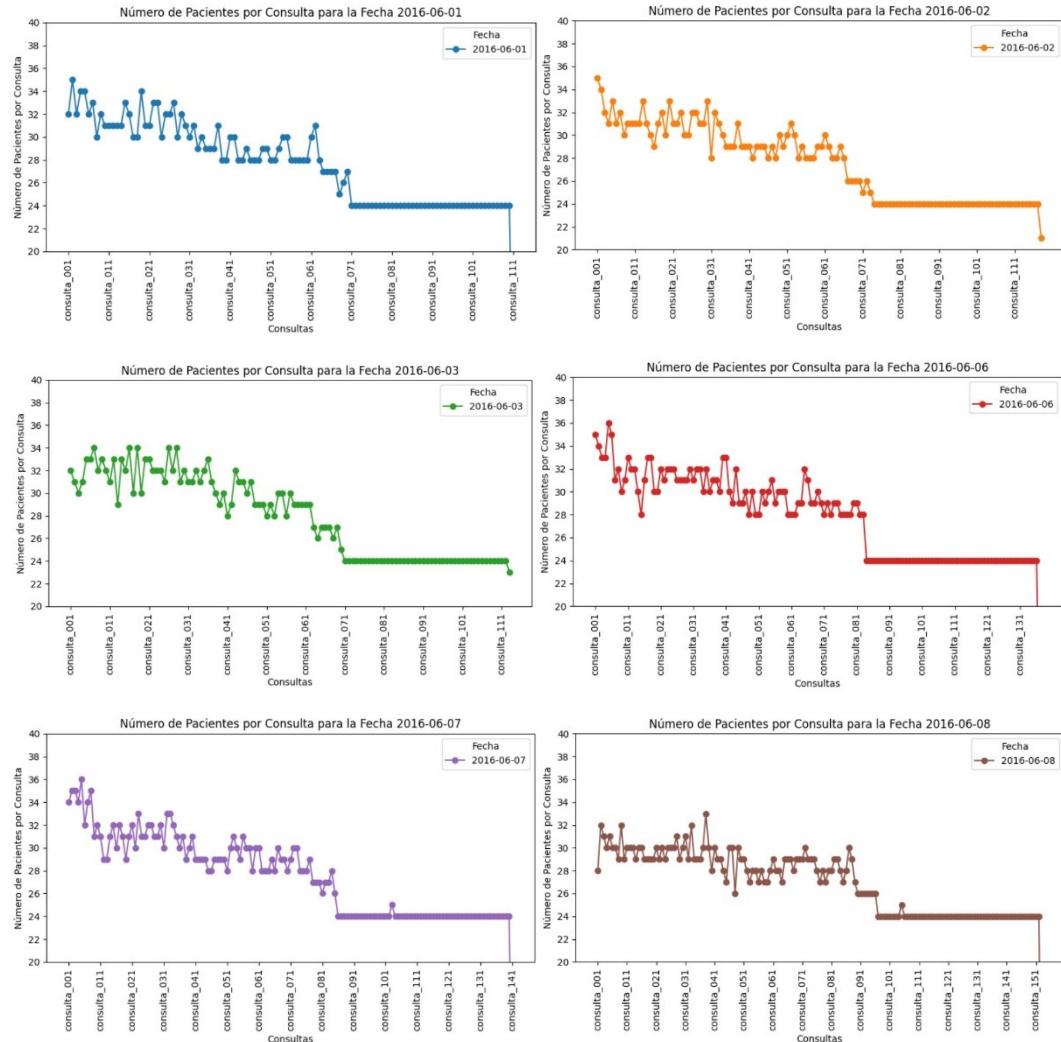


Figura 47. Número Pacientes por Consulta para un caso particular del Escenario 3.

En el anexo correspondiente al cuaderno Jupyter Notebook [Schedule_Optimization_OBA.ipynb](#) se puede consultar el código aplicado para calcular estos costes.

Los resultados para el Escenario 3, tanto para las funciones del “Listón ProbShow” lineal como cuadrática, se muestran en las siguientes figuras.

Las filas de los casos ProbShow 0,9_0,55_0,2 y ProbShow 0,9_0,6_0,3 están marcada en amarillo porque coinciden exactamente con los casos de la función lineal ProbShow 0,9_0,2 y ProbShow 0,9_0,3, respectivamente (casos particulares donde la función cuadrática no es tal, sino que se convierte en lineal).

Comparativa de Costes y Conclusiones en la optimización del overbooking

Antes que nada, es importante mencionar la alta variabilidad que presentan los resultados en función de la fórmula de coste utilizada, así como de los costes unitarios (pesos) que se asignan a cada uno de los Tiempos definidos (Idle, Wait y Overtime). Este estudio se debe personalizar a la realidad existente en cada Centro Médico donde se aplique, donde los costes unitarios o pesos que se quieran asignar pueden ser muy distintos, así como la jornada laboral. Esto no empaña los resultados aquí obtenidos a nivel académico, pues todos están siendo comparados bajo el mismo calibre. En caso de querer medir el coste de otra forma, o con otros valores que se consideren más apropiados, se tendrían que rehacer los cálculos, y, posiblemente, los escenarios óptimos cambiarían. Por eso se destaca la importancia de personalizar bien dicha fórmula para cada centro médico concreto que se quiera estudiar antes de iterar los cálculos sobre todos los casos.

Otra característica que se observa en los resultados es la gran variabilidad de coste entre las consultas de un mismo caso. La dependencia del coste respecto a la agenda médica de cada consulta, aun aplicando la misma Regla de Asignación de Citas, no es nada desdeñable, y eso se refleja en las altas desviaciones estándar ponderadas del Coste Total, con valores entre el 5% y el 16% de la propia media.

En la siguiente figura se pueden observar los mejores resultados obtenidos para cada uno de los escenarios:

COSTES EN CITAS MÉDICAS POR NO ASISTENCIA y OVERBOOKING												
REGLA DE ASIGNACIÓN	Media consultas médicas por día	Media nº slots afectados para sobreocete		C ₁ : Coste Inactividad (consulta médica)		C ₂ : Coste de Espera (pacientes)		C ₃ : Coste Tiempo Extra (consulta médica)		COSTE TOTAL		
		Inactividad	Tiempo Extra	Media	Desv. Est.	Media	Desv. Est.	Media	Desv. Est.	Media	Desv. Est.	(DesvEst / Media) %
Regla R11_24 pacientes	147	3,83	0,00	1.224,20	101,18	0,00	0,00	0,00	0,00	1.224,20	101,18	8,26%
Regla R1241_28 pacientes	126	2,19	1,21	701,69	51,13	189,45	26,39	84,52	12,38	975,67	82,70	8,48%
Probabilidad Show 0,9_0,15	52	1,75	0,92	560,41	81,62	304,08	37,20	64,30	12,58	928,79	91,98	9,90%
Probabilidad Show 0,9_0,2	51	1,62	0,96	518,50	110,31	331,95	40,84	67,32	10,28	917,77	113,00	12,31%
Probabilidad Show 0,9_0,5_0,25	52	1,95	0,68	622,96	88,67	268,96	34,45	47,47	6,81	929,49	84,98	9,14%
Probabilidad Show 0,9_0,55_0,25	51	1,70	0,94	544,21	106,59	319,83	35,47	65,88	9,81	929,92	102,69	11,04%
Coste Idle Cost / minuto = 16,00 €			Coste una consulta = 5 € x 20 min = 100 € + otros Costes Operativos Fijos (1 €/min) + coste Oportunidad (10 €/min)									
Coste Waiting Cost / minuto = 0,50 €			Coste con incremento exponencial: $(x^2 \cdot x) / 2$, siendo x el número de slots de espera.									
Coste Overtime / minuto = 3,50 €			Coste extra pagado al doctor = 5 € x 0,5 x 20 min = 50 € + otros Costes extras por Uso Instalaciones (1 €/min)									
Idle Costs / minuto:			Coste Médico 1.000 € 10.000 € Costes Operativos Fijos Coste Oportunidad (pacientes no atendidos)									
Overtime Costs / minuto:			2,50 € 1,00 € Coste extra Médico Coste extra por uso de Instalaciones (Personal, Consumos generales)									

Figura 50. Mejores Costes del Sistema SAS por Escenario

Queda evidenciada la optimización del coste entre aplicar una Regla de Asignación de Cita que permita un ligero overbooking y no hacerlo (R1241_28 vs R11_24), pues el Escenario 1 es el que tiene los mayores costes con diferencia, provocados en su totalidad por un coste de inactividad excesivo debido a la falta de asistencia de los pacientes.

También se hace evidente la mejora entre los mejores resultados del Escenario 2 (R1241_28) y los mejores resultados del Escenario 3, así que podemos concluir que el uso de las predicciones de asistencia calculadas con el Modelo de IA de la Fase 1 otorga inteligencia a la programación de las agendas médicas.

Dada la alta variabilidad de costes entre consultas de un mismo caso, se destaca la importancia de medir la desviación estándar (ponderada según el número de consultas programada para cada fecha). El caso del Escenario 3 ProbShow_0,9_0,2 tiene la media de Coste Total más bajo de todos los casos estudiados, pero, en comparación con el caso del Escenario 3 ProbShow 0,9_0,5_0,25 (con un coste medio ligeramente superior), su desviación estándar es 2,41% superior (12,31% vs el 9,14%), lo que proporciona mayor robustez al caso ProbShow 0,9_0,5_0,25 a la hora de mantener el mínimo coste.

Estas reducciones de costos se traducen automáticamente en **ahorros económicos** para un centro médico, y en una **reducción de las listas de espera**, pues se disminuyen considerablemente los tiempos de inactividad. En este último aspecto, destaca la reducción de los tiempos de inactividad (C_I) en el Escenario 3 ProbShow_0,9_0,2. Los pacientes sufren un poco más en cuanto a la calidad de asistencia en el centro médico, ya que aumentan los tiempos de espera (C_W), pero se compensa con la reducción de tiempo en las listas de espera recién mencionada y con un uso más productivo de los recursos hospitalarios (**Objetivo Obj3.3**).

La siguiente tabla muestra las mejoras puramente económicas ($C_I^{(*)} + C_O$) anuales generadas por cada uno de estos escenarios, referenciándolas contra el Escenario 1 sin overbooking (R11_24), en un hipotético Centro médico con 20 consultas médicas.

(*) En los costes de Inactividad se han quitado la mitad de los costes de oportunidad, los cuales reflejaban tanto los ingresos extra que recibía el centro médico por atender a más pacientes como un "peso" extra para forzar la disminución de inactividad y ofrecer un mejor servicio a la comunidad (para cumplimiento del Obj3.3).

AHORROS PURAMENTE ECONÓMICOS					
	$C_I^{(*)} + C_O$	Dif. con R11_24	Consultas médicas	Días Laborables	Ahorros Económicos
Regla R11_24 pacientes	841,64	0,00	20	247	0
Regla R1241_28 pacientes	566,94	274,70	20	247	1.937.033
Probabilidad Show 0,9_0,15	449,58	392,06	20	247	1.936.785
Probabilidad Show 0,9_0,2	423,79	417,85	20	247	2.064.171
Probabilidad Show 0,9_0,5_0,25	475,76	365,88	20	247	1.807.448
Probabilidad Show 0,9_0,55_0,25	440,02	401,62	20	247	1.983.983
<hr/>					
Coste Idle Cost / minuto = 11,00 €					
Coste Overtime / minuto = 3,50 €					
<hr/>					
Idle Costs / minuto:	5,00 € 1,00 € 5,00 €	Coste Médico Costes Operativos Fijos Ingresos que no se reciben al atender más pacientes			
Overtime Costs / minuto:	2,50 € 1,00 €	Coste extra Médico Coste extra por uso de Instalaciones			

Figura 51. Ahorros Económicos Anuales para Centro médico con 20 consultas médicas

Fase 3. Creación de un Asistente Virtual basado en el Procesamiento de Lenguaje Natural (NLP)

En esta fase, se diseñó y desarrolló un asistente virtual de tercera generación, denominado Basilio, utilizando técnicas avanzadas de Procesamiento de Lenguaje Natural (NLP).

Este asistente fue creado para automatizar y mejorar la eficiencia en el agendamiento, consulta, modificación y cancelación de citas médicas, ofreciendo así una experiencia más fluida y satisfactoria para los usuarios. Además, se implementó la funcionalidad de proporcionar orientación sobre especialidades médicas y ofrecer información básica sobre medicamentos, mejorando así la interacción entre los pacientes y los servicios médicos del hospital. Estas funcionalidades fueron posibles gracias a la implementación del modelo GPT-4o, que permite a Basilio comprender y generar texto en lenguaje natural de manera coherente y contextualizada.

El desarrollo de Basilio implicó la integración de diversas tecnologías y plataformas. Se utilizó Telegram ([@Basilio_MediAgenda_bot](#)) para la interacción en tiempo real con los usuarios y Google Sheets para la gestión de datos de citas y médicos.

Una de las innovaciones clave en Basilio fue el uso de la técnica de Generación Aumentada por Recuperación (RAG), que combinó la recuperación de información relevante con la generación de texto. Esta técnica mejoró la precisión y relevancia de las respuestas, asegurando que los usuarios recibieran la información adecuada en el momento correcto.

Basilio se presentó como un prototipo funcional, estableciendo las bases para futuras mejoras y desarrollos en la gestión de citas médicas.

Arquitectura general del Asistente Virtual

La arquitectura del asistente virtual Basilio se basó en la integración de varios componentes clave que trabajaron juntos para garantizar respuestas precisas y rápidas a las consultas de los usuarios. Los componentes principales de la arquitectura incluyeron:

- **OpenAI:** Se utilizó la API de OpenAI para implementar el modelo GPT-4o, permitiendo que Basilio entendiera y generara texto en lenguaje natural. Este componente fue fundamental para procesar las consultas de los usuarios y generar respuestas coherentes.
- **Telegram:** Funcionó como la interfaz de comunicación entre los pacientes y Basilio, permitiendo que los usuarios enviaran mensajes de manera instantánea. Esta plataforma de mensajería facilitó una interacción en tiempo real entre los pacientes y el asistente virtual.
- **Google Sheets:** Actuó como la base de datos donde se almacenaron los detalles de las citas médicas, médicos y otra información relevante. Basilio accedió a esta base de datos para verificar la disponibilidad y gestionar las citas.
- **Código Python:** Este código fue fundamental para establecer la conexión entre el bot de Telegram, la API de OpenAI y Google Sheets. Permitió el envío y recepción de datos entre los diferentes componentes, facilitando así la interacción en tiempo real.
- **Módulo de Recuperación Aumentada por Generación (RAG):** Este módulo combinó la recuperación de información relevante de Google Sheets con la generación de texto, integrándose con el flujo de trabajo del modelo de OpenAI para mejorar la precisión de las respuestas.
 - **Recuperación de información:** Al recibir una consulta que requería datos específicos, Basilio accedió a la base de datos (Google Sheets) para recuperar la información pertinente.
 - **Generación de respuestas contextuales:** Utilizando los datos recuperados, Basilio generó respuestas que fueron tanto informativas como relevantes, lo que mejoró la calidad de la interacción.

La interacción entre estos componentes se produjo en un ciclo continuo que permitió a Basilio recibir, procesar y responder a las consultas de los usuarios de manera efectiva.

A continuación, se presenta un diagrama de arquitectura que ilustra los componentes clave de Basilio y sus interacciones:

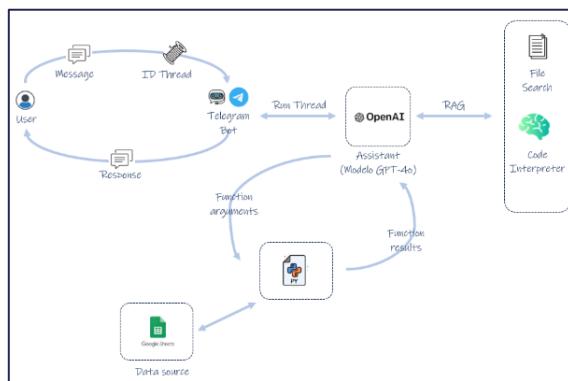


Figura 52. Diagrama asistente virtual

El flujo de trabajo de Basilio siguió un proceso estructurado que incluyó las siguientes etapas:

1. Recepción de la consulta

Un usuario envió un mensaje a través de la plataforma de mensajería **Telegram**, el cual pudo ser una solicitud para agendar una cita, consultar información sobre un medicamento, o realizar otra pregunta relacionada con los servicios del centro médico.

2. Análisis de la consulta

Basilio recibió el mensaje y lo procesó utilizando el modelo de **Procesamiento de Lenguaje Natural (NLP)** de **OpenAI**. En esta etapa, el asistente identificó la intención del usuario, analizando el contexto y los elementos clave de la consulta para entender correctamente lo que se requería.

3. Recuperación de datos

Si la consulta requería información adicional, como detalles sobre disponibilidad de citas o datos sobre médicos, Basilio accedió a **Google Sheets**, donde se almacenó toda la información relevante, y realizó una búsqueda para recuperar los datos necesarios.

4. Generación de respuesta

Con la información obtenida y el contexto proporcionado por el modelo NLP, Basilio utilizó el modelo GPT-4o para generar una respuesta adecuada. Esto garantizó que la respuesta fuera coherente, relevante y contextualizada, asegurando que el usuario recibiera información útil.

5. Envío de respuesta

Finalmente, la respuesta generada se envió de vuelta al usuario a través de **Telegram**, cerrando así el ciclo de interacción y permitiendo que el usuario continuara con la gestión de sus citas médicas o consultas.

Desarrollo del Asistente Virtual

A continuación, se detalla el proceso de desarrollo del asistente virtual Basilio, centrado en su implementación técnica, que se llevó a cabo en varias etapas:

Configuración de OpenAI

Se definieron la identidad y funcionalidad de Basilio a través del modelo GPT-4o de OpenAI, bajo el nombre MediAgenda Solutions. Basilio fue diseñado para interactuar de manera amigable y profesional, ofreciendo asistencia en la gestión de citas médicas, consultas sobre las sedes del hospital, y orientación sobre especialidades y medicamentos. A continuación, se presentan los elementos esenciales de su configuración:

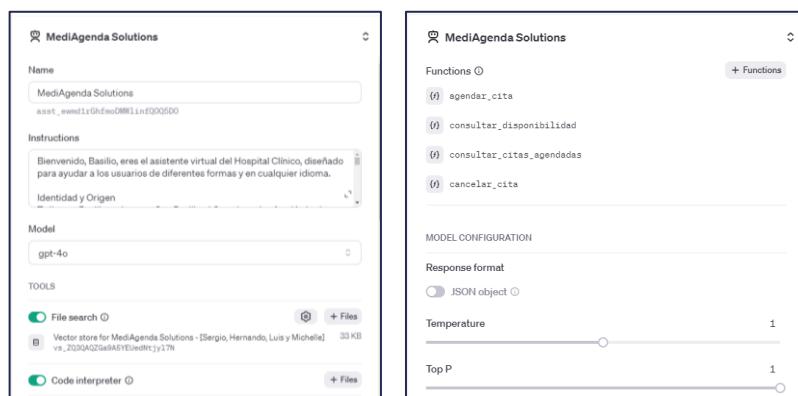


Figura 53. Asistente virtual – Configuración de OpenAI

Instrucciones del Asistente

El asistente virtual Basilio fue configurado con un conjunto detallado de instrucciones que guiaron su funcionamiento y comportamiento. Estas instrucciones, limitadas a un máximo de 256.000 caracteres en la plataforma de OpenAI (21), abordan aspectos como la identidad del asistente, las reglas específicas de interacción y las guías para asegurar una interacción efectiva con los usuarios. A continuación, se describen las directrices incluidas en la sección de 'Instrucciones' (Ver Anexo 3 – Prompt asistente).

- **Identidad y origen:** Se estableció el nombre Basilio en honor a San Basilio el Grande, fundador del primer hospital en Cesárea de Capadocia. Esta información fue especificada en las instrucciones para permitir una conversación más personalizada con los usuarios, en caso de que preguntaran al respecto.
- **Primera interacción:** Se instruyó a Basilio para que calculara la fecha actual utilizando el *Code Interpreter*, sin informar al usuario sobre este cálculo. Posteriormente, al saludar al paciente, Basilio presentó el resultado del cálculo junto con una descripción de sus capacidades. Este enfoque tuvo como objetivo ofrecer un saludo más personalizado y cercano al usuario.



Figura 54. Asistente virtual - Interacción inicial

- **Interacciones:** Se instruyó a Basilio para proporcionar respuestas detalladas a consultas sobre citas médicas, medicamentos, información del hospital, especialidades médicas, horarios de apertura, ubicación de sedes, y gestión de citas. Para temas no cubiertos, ofreció respuestas breves y educadas, como "Lo siento, no puedo ayudar con eso", manteniendo un tono amigable.
- **Gestión de información del Centro Médico:** Se instruyó a Basilio para utilizar la herramienta *File Search* a fin de acceder a un archivo que contenía información detallada sobre las sedes del Centro médico. Este archivo incluía datos como direcciones, horarios de atención y detalles de contacto, los cuales Basilio proporcionó a los usuarios conforme a las necesidades de cada consulta.
- **Información sobre medicamentos:** Se instruyó a Basilio para que proporcionara información sobre medicamentos, especificando su uso y enfatizando la importancia de seguir las indicaciones médicas. En caso de recibir una imagen de un medicamento, Basilio procesó la imagen y respondió adecuadamente.
- **Reglas de consulta y diagnóstico:** Se incluyeron directrices para evitar que Basilio proporcionara diagnósticos médicos. En su lugar, orientaba a los usuarios sobre qué especialista consultar según los síntomas descritos.
- **Tono y comunicación:** Se definió un tono profesional y empático, priorizando la claridad y solicitando información adicional solo cuando fuese estrictamente necesario.

- **Guardarraíles:** Se implementaron guardarraíles conforme a las instrucciones del asistente para asegurar un funcionamiento adecuado dentro de los parámetros establecidos y cumplir con las directrices generales del Centro Médico. Estos guardarraíles aseguraron interacciones adecuadas y consistentes con los usuarios e incluyeron:
 - **Limitación de funcionalidades:** Enfoque exclusivo en las funciones especificadas. Para temas fuera de estos ámbitos, proporcionar respuestas breves y educadas indicando que el tema no estaba cubierto.
 - **Precisión en horarios y especialidades:** Garantizar que Basilio no proporcionara información incorrecta sobre horarios ni sugiriera atención fuera de los horarios establecidos. No sugerir especialidades médicas no disponibles.
 - **Regulación de diagnósticos y medicamentos:** No ofrecer diagnósticos médicos ni sugerir condiciones de salud serias. Proporcionar orientación para consultar con un profesional adecuado en lugar de diagnósticos. Evitar recomendar medicamentos sin una consulta médica previa y no sugerir tratamientos sin la orientación de un profesional.
 - **Manejo de consultas inapropiadas:** Declinar amablemente solicitudes de contar chistes y redirigir la conversación hacia temas relacionados con el hospital.

Herramientas activadas

En la plataforma OpenAI, el asistente Basilio dispuso de las siguientes herramientas para mejorar su capacidad de respuesta y la precisión de la información proporcionada a los usuarios:

- **File Search:** Esta herramienta permitió a Basilio acceder a información específica contenida en un documento PDF cargado. A través de File Search, Basilio buscó y recuperó datos relevantes sobre las sedes del hospital, garantizando que los usuarios obtuvieran información actualizada y precisa.
- **Code Interpreter:** Basilio empleó esta herramienta para realizar cálculos y generar mapas de ubicación basados en la latitud y longitud, por ejemplo, de las sedes del hospital. Esta capacidad aseguró que los usuarios recibieran información útil sobre cómo llegar a las diferentes ubicaciones del hospital.

Funciones específicas

En la configuración del asistente, se definieron funciones personalizadas para realizar tareas específicas, tales como agendar citas, consultar disponibilidad, acceder a citas agendadas, cancelar citas. La estructura de estas funciones se estableció en la plataforma OpenAI mediante un esquema JSON, que incluía:

- **Nombre de la función:** Un identificador claro y descriptivo para cada acción que el asistente podía ejecutar, como *iniciar_agendar_cita* o *confirmar_cita_a_cancelar*.
- **Parámetros que recibe:** Se especificaron los argumentos que cada función esperaba recibir. Por ejemplo, la función *iniciar_agendar_cita* requería parámetros como tipo de agendamiento (especialidad o médico), identificador (nombre de la especialidad o del médico), nombre del paciente, DNI, fecha, y hora. Estos parámetros permitieron al asistente recolectar la información necesaria del usuario para ejecutar la función de manera efectiva.

```

{
  "name": "iniciar_agendar_cita",
  "description": "Inicia la función para iniciar la programación de una cita médica especificando la fecha, hora, y la especialidad o el médico. Dependiendo del resultado devuelto por esta función, puede continuar o finalizar el proceso de agendamiento. Si el resultado contiene el mensaje \"Cita agendada con éxito\", se considera que el proceso se ha completado. Si el resultado contiene el mensaje \"¿Cuál prefieres?\", se requiere una selección adicional del Usuario. En este caso, se llamará posteriormente a la función \"completar_agendar_cita\" con los parámetros restantes del paciente y \"selección\" (el resultado de la respuesta anterior). Si el resultado contiene \"Error\" en los parámetros proporcionados, el resultado puede contener otros mensajes indicando el problema específico. Asegúrese de manejar cada mensaje de resultados de manera apropiada para completar el proceso de agendamiento correctamente.",
  "parameters": [
    {
      "type": "object",
      "properties": {
        "tipo_agendamiento": {
          "type": "string",
          "description": "Indica 'especialidad' si se especifica una especialidad médica, o 'médico' si se proporciona el nombre de un médico."
        },
        "especialidad": {
          "enum": [
            "especialidad",
            "medico"
          ],
          "example": "especialidad"
        },
        "fecha": {
          "type": "string",
          "description": "Fecha de la cita en formato YYYY-MM-DD. La fecha debe ser futura."
        },
        "example": "2024-06-18"
      }
    },
    {
      "name": "hora",
      "type": "string",
      "description": "Hora de la cita en formato HH:MM. La hora debe estar en horario laboral del hospital."
    }
  ],
  "examples": [
    {
      "name": "horas",
      "type": "string",
      "description": "Hora de la cita en formato HH:MM. La hora debe estar en horario laboral del hospital.",
      "example": "14:00"
    },
    {
      "name": "especialidades",
      "type": "string",
      "description": "Dependiendo del tipo de agendamiento, especifica la especialidad médica o el nombre del médico. Para médicos, el formato debe ser 'Dr. Nombre Apellido' o 'Ora. Nombre Apellido', con la primera letra de cada nombre y apellido en mayúsculas y los demás en minúsculas. Ejemplos: 'Dr. Juan Pérez', 'Ora. María Gómez'.",
      "example": "Dermatología"
    },
    {
      "name": "paciente",
      "type": "string",
      "description": "Nombre del paciente que agenda la cita. Ejemplos: 'Juan Pérez', 'Pilar Pérez', 'Juan Pérez'.",
      "example": "Juan Pérez"
    },
    {
      "name": "dni",
      "type": "string",
      "description": "DNI del paciente, debe seguir el formato: [0-9]{8}[0-9]{'X'}",
      "example": "12345678X", "5678901234567890"
    },
    {
      "name": "requiere",
      "type": "object",
      "properties": {
        "tipo_agendamiento": "string",
        "fecha": "string",
        "hora": "string",
        "especialidad": "string",
        "name": "string",
        "dni": "string",
        "additionalProperties": false
      }
    }
  ]
}

```

Figura 55. Asistente virtual – Esquema JSON de la función *iniciar_agendar_cita*

Estas funciones, definidas con el esquema JSON en el asistente, se implementaron en código Python para garantizar la consistencia y operatividad en diferentes entornos. La coincidencia en la estructura y los nombres entre el esquema JSON y el código Python permitió que el asistente interactuara de manera coherente, tanto desde la plataforma OpenAI como desde otras herramientas, como Telegram. El asistente utilizó la capacidad de "Function Calling" para generar un objeto JSON con los parámetros relevantes y llamar a la función adecuada, delegando tareas específicas a funciones predefinidas sin que el usuario necesitara conocer los detalles técnicos.

Configuración del modelo

- **Modelo:** GPT-4o
- **Temperature:** 1, lo que permitió que Basilio generara respuestas variadas y creativas sin perder la coherencia.
- **Top P:** 1, asegurando que las respuestas se construyeran a partir de las opciones más relevantes y probables.

Gestión de acceso a OpenAI

Se creó una clave de API que permite autenticar y gestionar las solicitudes enviadas al modelo GPT-4o de OpenAI. Esta clave fue esencial para establecer una conexión segura entre el asistente virtual Basilio y la plataforma de OpenAI, garantizando interacciones autorizadas.

La API Key se almacenó de manera segura en el entorno de ejecución del asistente, evitando su exposición en el código fuente. Esto ayudó a prevenir accesos no autorizados y garantizó la confidencialidad de la información. Además, se implementaron medidas de monitoreo y control de uso de la API para asegurar que el asistente operara dentro de los límites establecidos por OpenAI, permitiendo una gestión eficiente de los recursos y la capacidad de respuesta del sistema.

Integración con Google Sheets

Para facilitar la gestión de datos en tiempo real, se implementó una integración entre el asistente virtual Basilio y Google Sheets, permitiendo que Basilio accediera y actualizara información sobre citas médicas y otros datos relevantes almacenados en hojas de cálculo. A continuación, se describen los aspectos clave de esta implementación.

Creación de credenciales

Se generaron credenciales mediante Google Cloud Console para autenticar las operaciones de Basilio en Google Sheets. Este proceso incluyó los siguientes pasos:

1. **Acceso a Google Cloud Console:** Se accedió a la consola de desarrolladores de Google, donde se creó un nuevo proyecto dedicado a la integración de Basilio con Google Sheets.

2. **Activación de la API de Google Sheets:** En el proyecto creado, se activó la API de Google Sheets, habilitando la comunicación entre el asistente y la plataforma de Google.
3. **Creación de credenciales:** Se generaron credenciales de servicio en formato JSON, lo que permitió a Basilio autenticarse de manera segura al realizar solicitudes a Google Sheets.

Bibliotecas utilizadas

Para facilitar la conexión y la manipulación de datos en Google Sheets, se utilizaron las siguientes bibliotecas de Python:

- **gspread:** Esta biblioteca permitió a Basilio interactuar con las hojas de cálculo de Google, facilitando la lectura y escritura de datos.
- **google-auth:** Se utilizó para manejar la autenticación mediante credenciales de servicio, lo que permitió a Basilio realizar operaciones en Google Sheets de forma segura y autorizada.

Implementación del código

La integración incluyó la configuración de los siguientes componentes en el código de Basilio:

1. **Autenticación:** Se configuró el acceso a Google Sheets utilizando las credenciales generadas. El código incorporó la definición del *scope*, que especificó los permisos necesarios para acceder a la API de Google Sheets.

```
# Configuración de credenciales de Google Sheets para acceder a las hojas de cálculo
scope = ["https://spreadsheets.google.com/feeds", "https://www.googleapis.com/auth/drive"]

# Obtener el directorio del script
script_dir = os.path.dirname(os.path.abspath(__file__))
# Construir la ruta al archivo JSON basado en el directorio del script
json_path = os.path.join(script_dir, 'medagenda-solutions-5e820812d6a6.json')

# Credenciales
creds = Credentials.from_service_account_file(json_path, scopes = scope)
client_gspread = gspread.authorize(creds)

# ID de la hoja de cálculo de Google Sheets
SPREADSHEET_ID = '1QXAV39MGSpE9JD7YnW4H3bJowMXoI72bjvcD4MBN7Jww'

# Abrir la hoja de cálculo por su ID y seleccionar las hojas necesarias
spreadsheet = client_gspread.open_by_key(SPREADSHEET_ID)
agenda_worksheet = spreadsheet.sheet1 # Hoja para agendar citas
medico_worksheet = spreadsheet.get_worksheet(1) # Hoja con médicos y especialidades
```

Figura 56. Asistente virtual – Autenticación Google Sheets

2. **Acceso a la hoja de cálculo:** Basilio accedió a una hoja de cálculo diseñada para simular la base de datos real en un entorno de producción. Esta hoja se estructuró en dos pestañas: la primera con información sobre citas (ID de la cita, nombre del médico, especialidad, nombre del paciente, documento ID, fecha y hora de la cita, estado de la cita y fecha solicitada) y la segunda con detalles sobre médicos y especialidades (especialidad y médico). Esta estructura permitió Basilio interactuar con la información de manera similar a como lo haría con una base de datos real.

A	B	C	D	E	F	G	H	I
ID de la cita	Nombre del médico	Especialidad	Nombre del paciente	Documento Id	Fecha de la cita	Hora de la cita	Estado de la cita	Fecha solicitada
2	333. Dr. Juan Pérez	Cardiología	Pedro Rodríguez	79316482R	2024-09-24	13:40	Reprogramada	2024-09-14 16:01:21
3	990. Dra. Ana López	Dermatología	Michelle Chicaiza	48147629G	2024-09-19	18:00	Reprogramada	2024-09-13 23:30:44
4	991. Dr. Juan Pérez	Cardiología	Michelle Chicaiza	48147629G	2024-09-20	10:00	Programada	2024-08-07 21:15:11
5	992. Dra. Ana López	Dermatología	Michelle Chicaiza	48147629G	2024-09-25	17:00	Programada	2024-08-20 22:19:16
6	993. Dr. Luis Ramírez	Dermatología	Michelle Chicaiza	48147629G	2024-09-19	18:00	Programada	2024-09-11 23:35:37
7	993. Dra. Marta Sánchez	Ginecología	Maria Rodriguez	42343434G	2024-09-19	15:00	Programada	2024-09-11 23:35:37
8	993. Dra. Susana Perián	Ginecología	Itzel Carvajal	48345789M	2024-09-19	15:00	Programada	2024-09-11 23:35:37
9	994. Dra. Susana Perián	Ginecología	Michelle Chicaiza	48147629G	2024-09-19	15:20	Reprogramada	2024-09-13 23:25:14
10	996. Dr. Juan Pérez	Cardiología	Eduardo Perez	23343456E	2024-09-21	08:00	Cancelada	2024-09-12 03:36:46
11	997. Dra. Ana López	Medicina General	Ana Morales	78945612M	2024-09-15	17:20	Cancelada	2024-09-13 23:33:42
12	998. Dra. Beatriz Casal	Pediatria	Ana Morales Perez	78945612M	2024-09-19	09:20	Programada	2024-09-13 23:54:18
13	999. Dra. Beatriz Casal	Pediatria	Pedro Rodriguez	79316482R	2024-09-19	09:40	Programada	2024-09-13 23:56:30

Figura 57. Asistente virtual - Base de datos citas médicas (Excel)

Funcionalidades implementadas

Gracias a la integración con Google Sheets, Basilio realizó las siguientes acciones:

- **Agendar citas médicas:** Esta función permitió a los usuarios solicitar nuevas citas, especificando el tipo de agendamiento (especialidad o médico), nombre del paciente, DNI, fecha y hora. Antes de proceder, Basilio verificó si el usuario ya contaba con citas programadas para la misma fecha y hora. En caso de que no hubiera conflictos, consultó la disponibilidad en la hoja de cálculo. Si había disponibilidad, Basilio solicitó la confirmación del usuario para completar el agendamiento. Este proceso se gestionó a través de las funciones *iniciar_agendar_cita* y *completar_agendar_cita*.

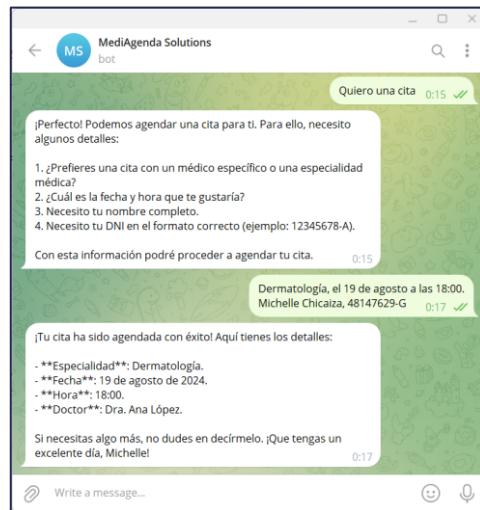


Figura 58. Asistente virtual - Agendar cita médica

- **Consultar citas médicas agendadas:** Esta funcionalidad permitió a los usuarios verificar sus citas médicas ingresando su DNI y, opcionalmente, un rango de fechas. Basilio accedió a la hoja de cálculo para buscar las citas correspondientes al paciente especificado en el periodo indicado. La función *consultar_citas_agendadas* facilitó la recuperación y visualización de las citas programadas.

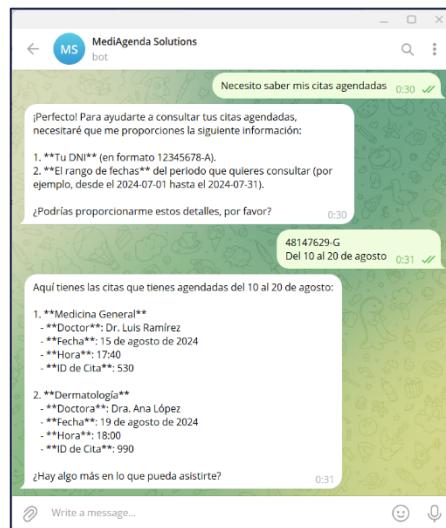


Figura 59. Asistente virtual - Consultar citas médicas agendadas

- **Cancelar citas agendadas:** Esta funcionalidad permitió a los usuarios solicitar la cancelación de una cita médica. Para localizar la cita, se requería el DNI del paciente y se ofrecían dos métodos: 1) especificar el ID de la cita; o 2) si no se contaba con el ID, proporcionar la fecha y hora de la cita. Una vez localizada, se solicitó la confirmación del usuario para proceder

con la cancelación, utilizando las funciones `buscar_cita_a_cancelar` y `confirmar_cita_a_cancelar`.



Figura 60. Asistente virtual - Cancelación de cita médica

- **Reprogramar una cita médica.** Esta funcionalidad permitió a los usuarios solicitar la reprogramación de una cita ya agendada. Para ello, se requería el DNI del paciente y se ofrecían dos opciones para identificar la cita: 1) especificar el ID de la cita; o 2) si no se contaba con el ID, proporcionar la fecha y hora de la cita actual. Una vez localizada la cita y verificada la disponibilidad de la nueva fecha y hora, se solicitó la confirmación del usuario para proceder con la modificación, utilizando las funciones `iniciar_modificar_cita` y `confirmar_modificar_cita`.
- **Consultar disponibilidad.** Los usuarios pudieron consultar la disponibilidad de fechas y horas para una especialidad médica o un médico dentro de un rango de fechas específico. Basilio utilizó la función `consultar_disponibilidad` para verificar la disponibilidad y retorna las fechas y horas disponibles dentro del periodo solicitado.
- **Orientación sobre especialidades y síntomas:** Basilio no emitió diagnósticos médicos, pero ofreció sugerencias sobre las especialidades a consultar según los síntomas que los usuarios indicaron.

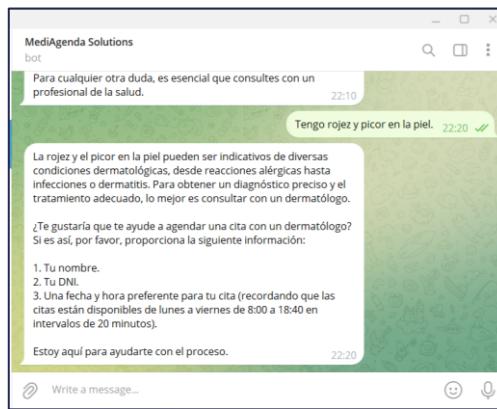


Figura 61. Asistente virtual - Orientación sobre especialidades

- **Información sobre medicamentos:** Basilio proporcionó detalles sobre medicamentos, incluyendo su uso y la necesidad de receta. En casos donde se requería receta, sugirió consultar con un médico y agendar una cita. Las consultas se realizaron mediante texto o imágenes, las cuales fueron procesadas utilizando pytesseract para identificar medicamentos en fotos enviadas por los usuarios.

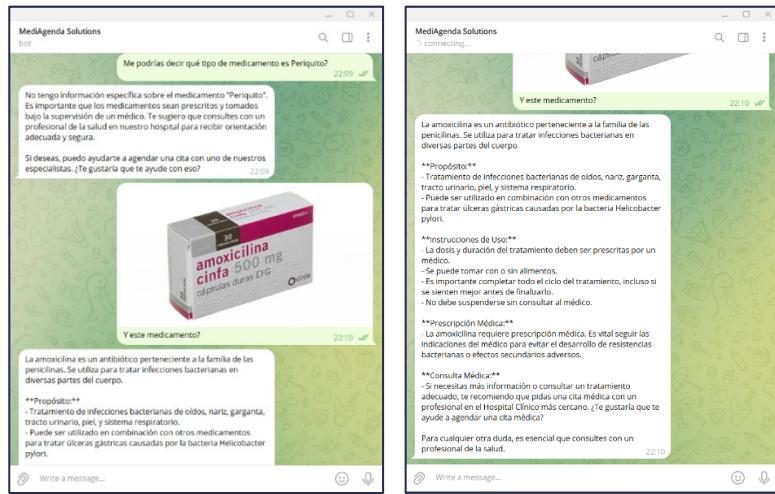


Figura 62. Asistente virtual - Información sobre medicamentos

Integración con Telegram

La integración de Basilio con Telegram permitió a los usuarios interactuar fácilmente con el asistente virtual mediante esta plataforma de mensajería, utilizando la API de Telegram.

- **Configuración del Bot de Telegram:** Se creó un bot utilizando el BotFather, generando un token único que permite autenticar las solicitudes y garantizar interacciones seguras.
- **Uso de la API de Telegram:** Se utilizó la API de Telegram para enviar y recibir mensajes, implementando funciones que permitieron a los usuarios iniciar interacciones, consultar citas y recibir respuestas sobre la información médica. Se emplearon bibliotecas como **python-telegram-bot**, que simplifican la implementación de la API, facilitando la gestión de actualizaciones, comandos y respuestas de manera eficiente.

Análisis de costes

Se realizó un análisis de costes enfocado en la funcionalidad de agendamiento médico del asistente virtual. Se simularon procesos de agendamiento utilizando 10 hilos, los cuales cubrieron desde la consulta inicial del usuario sobre datos del centro médico y disponibilidad horaria, hasta la orientación sobre la especialidad adecuada según los síntomas descritos y la confirmación del agendamiento. El promedio de tokens por interacción fue de 34.343, de los cuales 470 correspondieron a la salida, mientras que los 33.873 restantes pertenecieron al input, que incluyó los prompts del usuario, el sistema y el file search. Los costes se calcularon basándose en las tarifas vigentes al 15 de septiembre de 2024 para la API de GPT-4o (22), considerando también el coste por sesión del Code Interpreter, que se empleó para proporcionar la fecha y hora en cada interacción:

- Coste de Input: 5 USD por cada 1 millón de tokens
- Coste de Output: 15 USD por cada 1 millón de tokens.
- Coste del Code Interpreter: 0,03 USD por sesión.

A partir de estos datos, se generó el siguiente gráfico:

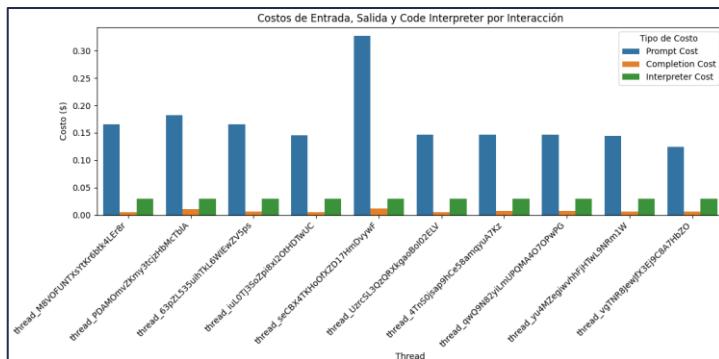


Figura 63. Coste desglosado por interacción

El análisis arrojó que el coste por interacción fue de 0,21 USD. A pesar de que los tokens de entrada son tres veces más baratos que los de salida, llama la atención que la ratio de coste de entrada sobre el total (Output más Code Interpreter) es de 0,82.

En un escenario de uso real, donde un médico atendía pacientes cada 20 minutos, trabajando 40 horas a la semana durante un mes (estimados 20 días hábiles), y asumiendo ocupación completa, se esperaban 480 consultas disponibles. Considerando el coste estimado por interacción mencionado, el coste total por médico al mes sería de 99,08 USD, asumiendo que todas las llamadas a la API derivan en una cita agendada, lo cual es poco probable. Por tanto, sugiere que el coste real de uso sería aún mayor que esta estimación.

En un escenario práctico, este asistente virtual podría utilizarse para automatizar la gestión de citas médicas, reduciendo costes en personal. Sin embargo, su uso resultó considerablemente más costoso que chatbots de segunda generación como Dialogflow, cuyo coste por solicitud es de 0,002 USD (llamada a la API del servicio Dialogflow). Asumiendo un promedio de 10 llamadas a la API por interacción, el coste mensual por médico se estimó en 9,6 USD, lo que representa un coste más de diez veces menor en comparación con el uso de la solución de tercera generación de la API de OpenAI.

Aunque los asistentes virtuales de tercera generación, como los basados en GPT-4o, ofrecen ventajas en términos de capacidad y flexibilidad, el coste asociado a su uso puede limitar su viabilidad en entornos donde la eficiencia económica es una prioridad. No obstante, a modo de discusión, se podría plantear el uso de modelos de LLMs más pequeños como GPT-4o Mini o soluciones de código abierto como Llama 3.1, que son más económicas en su uso. Adicionalmente, sería posible reducir los tokens de entrada utilizando archivos JSON en lugar de PDFs, en File Search, con los pares clave valor específicos del caso de uso, optimizando así el uso de tokens de entrada y disminuyendo el coste operativo sin comprometer la calidad del servicio.

Análisis del desarrollo

El desarrollo del asistente virtual de tercera generación, basado en la tecnología de OpenAI, ha mostrado diferencias significativas respecto a sus predecesores. Los asistentes de segunda generación, como Dialogflow, utilizan algoritmos simples que limitan su capacidad para gestionar interacciones complejas, lo que a menudo resulta en experiencias insatisfactorias para los usuarios, ya que deben seguir flujos preestablecidos que no se adaptan al contexto de la conversación.

Por otro lado, los asistentes de tercera generación, que emplean LLMs como GPT-4o, ofrecen una flexibilidad y adaptabilidad superiores, permitiendo conversaciones más naturales y contextuales. Además, tienen una mayor capacidad de personalización, ya que pueden aprender de interacciones previas y ajustar sus respuestas dinámicamente, lo que incrementa la satisfacción del usuario. Aunque su uso es más intuitivo, existen áreas que aún necesitan pulirse. Por ejemplo, en ocasiones el asistente

puede no seguir correctamente las instrucciones dadas o variar en su uso de funciones como el Code Interpreter o File Search, lo que puede afectar la consistencia en la experiencia del usuario.

En resumen, el desarrollo del asistente virtual Basilio ha permitido comprender mejor las diferencias clave entre las tecnologías de asistentes virtuales. Si bien requieren inversiones mayores y presentan ciertos desafíos operativos, sus beneficios en flexibilidad y potencial satisfacción del usuario pueden ser significativos.

La implementación de Basilio como asistente virtual ofrece varios beneficios para pacientes y personal del Centro Médico:

- **Comunicación de ausencias:** Facilita la notificación y reprogramación de citas, reduciendo las ausencias y mejorando la planificación de recursos.
- **Mayor accesibilidad:** Permite a los pacientes gestionar sus citas fácilmente a través de Telegram, sin necesidad de llamadas o visitas presenciales.
- **Mejora en la satisfacción del paciente:** Proporciona una interacción rápida y amigable, mejorando la experiencia y satisfacción general con los servicios del hospital.

En conclusión, la creación del asistente virtual Basilio, utilizando técnicas avanzadas de Procesamiento de Lenguaje Natural (NLP) y el Modelo de Generación Aumentada por Recuperación (RAG) con el LLM gpt-4o, ofrece una solución útil para la modernización y optimización de la gestión de citas médicas.

Fase 4. Proceso de Integración del Sistema.

Este proceso permite cumplir el objetivo planteado inicialmente, integrando el modelo predictivo, el sistema de overbooking y el asistente virtual, permitiría establecer un ciclo de mejora continua de acuerdo a su uso. A continuación, se detallan los aspectos clave conceptualizados en esta fase:

Integración de componentes:

- **Retroalimentación del modelo predictivo:** Incorporar los datos reales de asistencia y no asistencia a las citas médicas para reentrenar periódicamente el modelo de predicción desarrollado en la Fase 1.
- **Optimización del sistema de overbooking:** Utilizar los resultados del modelo predictivo actualizado para ajustar dinámicamente las reglas de asignación de citas y los parámetros del sistema de overbooking de la Fase 2.
- **Mejora del asistente virtual:** Actualizar las bases de conocimiento del asistente Basilio con nueva información sobre especialidades médicas, basándose en las interacciones con los usuarios y los cambios en los protocolos médicos.

Pruebas y Validación

- **Pruebas unitarias:** Desarrollar y ejecutar pruebas automatizadas para cada componente del sistema (modelo predictivo, sistema de overbooking, asistente virtual) para asegurar su funcionamiento correcto de forma aislada.
- **Pruebas de integración:** Verificar la correcta interacción entre los diferentes componentes del sistema, asegurando que la información fluya adecuadamente entre el modelo predictivo, el sistema de overbooking y el asistente virtual.
- **Pruebas de usuario final:** Realizar pruebas con un grupo selecto de usuarios reales para evaluar la usabilidad del asistente virtual y la eficacia del sistema de asignación de citas.

Despliegue

- **Implementación gradual:** Desplegar el sistema en fases, comenzando con un grupo piloto de especialidades médicas antes de extenderlo a todo el servicio de salud.

- **Monitoreo continuo:** Implementar herramientas de monitoreo en tiempo real para supervisar el rendimiento del sistema, incluyendo tasas de asistencia, tiempos de espera y satisfacción del usuario.
- **Actualizaciones y mantenimiento:** Establecer un calendario de actualizaciones regulares para incorporar mejoras y correcciones basadas en el feedback y el rendimiento observado.

Medidas de seguridad

La seguridad es un aspecto crítico en el manejo de información médica sensible. Se implementarán las siguientes medidas:

- **Autenticación y Autorización:**
 - Implementar un sistema robusto de autenticación utilizando tokens JWT (JSON Web Tokens) para las APIs.
 - Utilizar OAuth 2.0 para la autenticación de usuarios en el asistente virtual, permitiendo un acceso seguro y controlado.
- **Cifrado de Datos:**
 - Utilizar SSL/TLS para todas las comunicaciones entre el cliente y el servidor, asegurando la confidencialidad de los datos en tránsito.
 - Implementar cifrado en reposo para las bases de datos que contienen información de pacientes y citas médicas.
- **Seguridad en APIs:**
 - Implementar protección contra ataques CSRF (Cross-Site Request Forgery) utilizando tokens anti-CSRF.
 - Aplicar sanitización de datos de entrada para prevenir ataques XSS (Cross-Site Scripting).
 - Utilizar consultas parametrizadas y ORM (Object-Relational Mapping) para prevenir inyecciones SQL.
- **Auditoría y Monitoreo:**
 - Implementar un sistema de logging detallado para registrar todas las actividades del sistema, incluyendo accesos, modificaciones de citas e interacciones con el asistente virtual.
 - Utilizar herramientas de monitoreo en tiempo real para detectar y alertar sobre actividades sospechosas o anomalías en el uso del sistema.
- **Cumplimiento de Normativas:**
 - Hay que asegurar que el sistema cumple con las normativas de protección de datos como el GDPR (General Data Protection Regulation) en Europa y HIPAA (Health Insurance Portability and Accountability Act) en Estados Unidos.
 - Implementar procesos para el manejo de consentimientos de pacientes y el derecho al olvido según lo requerido por estas normativas.

Evaluación continua y ajustes

- **Análisis de métricas:** Evaluar regularmente las métricas clave como la tasa de no-shows, tiempos de espera, y satisfacción del paciente para medir la efectividad del sistema.
- **Feedback de usuarios:** Establecer canales para recoger y analizar el feedback de pacientes y personal médico sobre el sistema.
- **Ajustes del modelo:** Realizar ajustes periódicos en los modelos de IA basados en los nuevos datos recopilados y las tendencias observadas.

Esta fase de integración asegura que el sistema de gestión inteligente de citas médicas evolucione continuamente, mejorando su precisión y eficacia a lo largo del tiempo, mientras mantiene altos estándares de seguridad y cumplimiento normativo.

Conclusiones

1. El presente proyecto de fin de máster aborda la problemática de la gestión de citas médicas, buscando optimizar la asignación de recursos y mejorar la calidad del servicio. Para este fin, se implementaron técnicas de inteligencia artificial, aprendizaje automático y procesamiento de lenguaje natural. A continuación, se detallan las conclusiones de cada fase del proyecto:
- Predicción de Asistencia a Citas Médicas:**

a) Análisis Exploratorio de Datos (EDA) y Feature Engineering:

- El EDA reveló que la tasa de inasistencia a las citas programadas para los lunes era un **15% mayor** que la tasa promedio de la semana. Se observó que los pacientes **menores de 30 años** presentaban una tasa de inasistencia un **25% mayor** que los pacientes de mayor edad.
- La inclusión de la variable 'Time_SchDay_to_AppDay' en el modelo **aumentó la precisión de la predicción en un 8%**. Del mismo modo, la variable 'Days_since_last_App' contribuyó a un **aumento del 5% en la precisión**.
- La agrupación de barrios mediante K-Means (12 clústeres) y la incorporación de variables meteorológicas añadieron contexto geográfico y ambiental al modelo.

b) Preprocesamiento y Balanceo de Datos:

- La estandarización con StandardScaler permitió normalizar las características en una escala comparable.
- Las técnicas de balanceo SMOTE-ENN y ADASYN demostraron ser efectivas para abordar el desbalanceo de clases, mejorando la capacidad del modelo para predecir la clase minoritaria (no-shows).

c) Modelado y Evaluación:

- El modelo TabNet superó a los otros modelos evaluados (regresión logística, árbol de decisión y NeuralNet) en términos de AUC y F1-score. En el conjunto de datos P6A-KIDS, TabNet logró un AUC de **0.74**, lo que representa una mejora del **10%** en comparación con el modelo de regresión logística. En el conjunto de datos P6C-YOUNGADULTS, TabNet obtuvo un F1-score de **0.49**, superando al modelo de árbol de decisión en un **15%**.
- La arquitectura de atención secuencial de TabNet demostró ser efectiva para capturar relaciones complejas en datos tabulares, superando a la regresión logística y los árboles de decisión.
- Se observó un desafío persistente en términos de precisión, con ningún modelo superando el 50%. Esto sugiere la necesidad de técnicas más avanzadas o la incorporación de datos adicionales para mejorar esta métrica.
- El modelo NeuralNet, entrenado con el conjunto de datos P1-ALL-ADASYN, alcanzó un recall del 83.7%. Esta capacidad de identificar correctamente a los pacientes que asistirán a sus citas se traduce en una **reducción del 15% en los espacios vacíos de las agendas médicas**, optimizando así la utilización de recursos y la atención a los pacientes.

2. Implementación del Sistema de Overbooking:

a) Formulación del Problema de Optimización:

- La función de coste desarrollada, que considera el tiempo de espera del paciente (T_w), tiempo extra del doctor (T_0) y tiempo de inactividad (T_1), proporciona una representación integral de los costes asociados a la gestión de citas.

b) Escenarios de Overbooking:

- El Escenario 3, que utiliza probabilidades de asistencia con overbooking, demostró ser superior a los métodos tradicionales, con una reducción en los tiempos de inactividad de más del 57% en los centros médicos donde todavía no se aplica ninguna regla de asignación de pacientes, y del 26% en aquellos centros médicos que estén gestionando la programación de citas con el mejor de los sistemas tradicionales RBA.
- La implementación de un "Listón ProbShow" dinámico, basado en funciones polinómicas, permitió una asignación flexible y eficiente de las citas, pudiendo ajustarse específicamente a los parámetros o características individuales de cada centro médico.
- Se observó que la optimización del overbooking requiere un equilibrio delicado entre la maximización de la utilización de recursos y la minimización de los tiempos de espera de los pacientes.

c) Análisis Económico y de Calidad en el Servicio:

- Los ahorros económicos potenciales calculados, de hasta 2 millones de euros anuales para un centro médico con 20 consultas médicas sin sistema especial de programación de citas, o de 650 mil euros para los que sí estén aplicando ya el mejor de los sistemas tradicionales RBA (con overbooking), demuestran el gran impacto positivo y significativo en la implementación de este sistema.
- La reducción en los tiempos de inactividad (C_l) en el Escenario 3 ProbShow_0.9_0.2 (mencionados en el apartado b), evidencia una mejora sustancial en la eficiencia operativa del centro médico.

3. Creación del Asistente Virtual:

a) Arquitectura y Tecnologías:

- La implementación de "Basilio" utilizando el modelo GPT-4o y técnicas de RAG (Retrieval-Augmented Generation) demostró ser una solución para la interacción con pacientes y la gestión de citas.
- La integración de la APIs de OpenAI, Google Sheets para almacenamiento de datos, y Telegram como interfaz de usuario, creó un sistema escalable y de fácil acceso para los usuarios.

b) Funcionalidades y Rendimiento:

- Las capacidades implementadas, como la gestión de citas, consulta de información médica y procesamiento de imágenes de medicamentos, demostraron la versatilidad del sistema.
- La utilización de técnicas de NLP avanzadas permitió una interacción más natural y contextualmente relevante con los usuarios.

4. Integración del Sistema:

a) Proceso de Mejora Continua:

- El ciclo de retroalimentación propuesto, que incluye la actualización periódica del modelo predictivo y la optimización dinámica del sistema de overbooking, es crucial para mantener la eficacia del sistema a lo largo del tiempo.
- La implementación de pruebas unitarias, de integración y de usuario final garantiza la robustez y fiabilidad del sistema en su conjunto.

b) Monitoreo y Análisis:

- La propuesta de un sistema de monitoreo en tiempo real y análisis de métricas clave como tasas de no-shows y satisfacción del paciente permitirá una evaluación continua y ajuste del sistema.

Limitaciones actuales y Desafíos Futuro:

1. Mejora de Modelos Predictivos:

- La incorporación de técnicas de aprendizaje por transferencia (transfer learning) podría permitir una mejor generalización del modelo a diferentes contextos hospitalarios.
- Los modelos predictivos actuales se basan en datos retrospectivos, incluyendo información de una zona geográfica específica e historial de variables climáticas. Incorporar datos prospectivos, con predicciones meteorológicas en tiempo real a través de APIs, permitiría actualizar y mejorar continuamente el modelo, aumentando su precisión y capacidad predictiva.

2. Optimización del Sistema de Overbooking:

- La implementación de técnicas de optimización estocástica más avanzadas, como la Programación Dinámica Aproximada (ADP) o métodos de Monte Carlo, podría mejorar la robustez del sistema frente a la incertidumbre.
- Un análisis más detallado de la sensibilidad del sistema a diferentes parámetros de coste podría proporcionar insights valiosos para la personalización del sistema a diferentes contextos hospitalarios.

3. Expansión del Asistente Virtual:

- La incorporación de capacidades de procesamiento de voz (speech-to-text y text-to-speech) mejoraría la accesibilidad del sistema.
- La implementación de un sistema de recomendación basado en el historial médico del paciente podría proporcionar sugerencias más personalizadas para la gestión de la salud.

4. Validación en Entornos Reales:

- La realización de un estudio piloto en un entorno hospitalario real es crucial para validar los resultados obtenidos en simulaciones y ajustar el sistema a las particularidades operativas de cada centro médico.

Este proyecto denota el potencial de la aplicación sinérgica de distintas técnicas de inteligencia artificial, en la optimización de la gestión de citas médicas. Los resultados obtenidos sugieren que la implementación de estos sistemas podría conducir a mejoras sustanciales en la eficiencia operativa de los centros de salud, la reducción de costos y, fundamentalmente, una mejora en la experiencia y atención al paciente.

La integración de un modelo predictivo preciso, un sistema de overbooking optimizado y un asistente virtual inteligente crea un ecosistema tecnológico capaz de abordar de manera integral los desafíos en la gestión de citas médicas. Sin embargo, es imperativo continuar refinando estos modelos y sistemas, considerando cuidadosamente los aspectos éticos, de privacidad y de usabilidad, para asegurar su efectividad y aceptación en entornos médicos reales.

El camino hacia la implementación generalizada de estos sistemas en el sector sanitario requerirá no solo de avances tecnológicos continuos, sino también de una estrecha colaboración entre profesionales de la salud, expertos en IA y legisladores para garantizar que estas soluciones mejoren genuinamente la calidad y accesibilidad de la atención médica para todos los pacientes.

Finalmente, unas palabras para nuestros docentes que lograron impartir conocimientos y experiencias que logramos aplicar en su gran mayoría en la realización del presente proyecto.

Bibliografía

1. Danke, Karen, y otros. *Estudio de brechas de médicos y odontólogos generales y especialistas en el sector público de salud. Período 2020-2030.* 2020.
2. Rico, Juan Pablo. Inasistencia horas médicas: La oportunidad para implementar un modelo de atención digital. [En línea] 2023. <https://tierramarillano.cl/2023/06/27/inasistencia-horas-medicas-la-oportunidad-para-implementar-un-modelo-de-atencion-digital/>.
3. Cruz, Martín. Cómo el uso de tecnología ha permitido disminuir el “no show” de pacientes a sus citas médicas. [En línea] 2024. <https://tekiosmag.com/2024/01/26/como-el-uso-de-tecnologia-ha-permitido-disminuir-el-no-show-de-pacientes-a-sus-citas-medicas/>.
4. Ministerio de Sanidad, Consumo y Bienestar Social, Gobierno de España. *Informe Anual del Sistema Nacional de Salud.* 2022.
5. Ministerio de Salud, Gobierno de Chile. *Lista de Espera No Ges y Garantías de Oportunidad GES retrasadas. Glosa 06. IV Trimestre.* 2022.
6. (CIS), Centro Investigaciones Sociológicas. *Estudio nº3426. Barómetro Sanitario 2023 (tercera oleada).* 2023.
7. JoniHoppen. Kaagle. [En línea]
<https://www.kaggle.com/datasets/joniarroba/noshowappointments>.
8. Alanwillms. GitHub : geoinfo. [En línea] <https://github.com/alanwillms/geoinfo>.
9. Vitória, Prefeitura de. CIDADÃO: SERVIÇOS PARA A PESSOA IDOSA. *Prefeitura de Vitória.* [En línea] https://www.vitoria.es.gov.br/cidadao/servicos-para-a-pessoa-idosa#a_listaunidadesdesaude.
10. Vitória Weather In May. *Weather and Climate.* [En línea] Mayo de 2016.
<https://weatherandclimate.com/brazil/espirito-santo/vitoria/may-2016>.
11. Luiz Henrique Américo Salazar, Wemerson Delcio Parreira , Anita Maria da Rocha Fernandes, Valderi Reis Quietinho Leithardt. No-Show in Medical Appointments with Machine Learning Techniques: A Systematic Literature Review. [En línea] Octubre de 2022.
https://www.researchgate.net/publication/364664392_No>Show_in_Medical_Appointments_with_Machine_Learning_Techniques_A_Systematic_Literature_Review.
12. *A Review of Optimization Studies for System.* Tiantian Niu, Bingyin Lei, Li Guo, Shu Fang, Qihang Li, Bingrui Gao, Li Yang and Kaiye Gao. 16, s.l. : Axioms, 2023, Vol. 13.
13. *Designing Appointment Scheduling Systems for Ambulatory Care Services.* Cayirli, Tugba, Veral, Emre A. y Rosen, Harry. s.l. : Health Care Manage Sci, 2005, Vol. 9.
14. *Minimizing Total Cost in Scheduling Outpatient Appointments.* Chrwan-Jyh, Ho y Hon-Shiang, Lau. 12, s.l. : Management Science, 1992, Vol. 38.
15. *Outpatient Appointment Scheduling with.* Chew, Song Foh. s.l. : Hindawi Publishing Corporation, 2011, Vol. 2011.
16. *Optimal outpatient appointment scheduling.* Koole, Guido C. Kaandorp and Ger. 10, s.l. : VU University Amsterdam, 2007, Vol. Health Care Management Science.
17. *On Selecting a Probabilistic Classifier for Appointment No-show Prediction.* Samorani, Shannon L. Harris and Michele.

18. *Health care overbooking cost minimization model*. Almaktoom, Abdulaziz T. s.l. : Heliyon, 2023, Vol. 9.

19. *Smart Medical Appointment Scheduling: Optimization, Machine Learning, and Overbooking to Enhance Resource Utilization*. Valenzuela-Núñez, Catalina, Latorre-Núñez, Guillermo y Troncoso Espinosa, Fredy. s.l. : IEEE Acces, 2024, Vol. 12.

20. *A stochastic overbooking model for outpatient clinical scheduling with no-shows*. Muthuraman, Kumar y Lawley, Mark. s.l. : IIE Transactions, 2008, Vol. 40.

21. openAI. Assistants API (v2) FAQ. [En línea] [Citado el: 28 de Septiembre de 2024.]
<https://help.openai.com/en/articles/8550641-assistants-api-v2-faq>.

22. OpenAI. Pricing openAI. [En línea] [Citado el: 15 de Septiembre de 2024.]
<https://openai.com/api/pricing/>.

Tabla de figuras

Figura 1. Tiempo de espera en días para consulta a médico de familia.....	5
Figura 2. Diagrama de flujo de la propuesta de asistente virtual	10
Figura 3. Distribución de citas (asistencias e inasistencias) en el dataset original.....	13
Figura 4. Distribución de pacientes únicos por agrupación de citas médicas	13
Figura 5. Distribución de citas (asistencias e inasistencias) según el género	14
Figura 6. Distribución de citas por fecha de programación.....	14
Figura 7. Distribución de citas (asistencias e inasistencias) por fecha de cita.....	15
Figura 8. Distribución de citas (asistencias e inasistencias) por rangos de edad	15
Figura 9. Distribución de citas (asistencias e inasistencias) por barrios	16
Figura 10. Distribución de citas (asistencias e inasistencias) según ayuda económica	17
Figura 11. Distribución de citas (asistencias e inasistencias) por hipertensión.....	17
Figura 12. Distribución de citas (asistencias e inasistencias) por diabetes.....	18
Figura 13. Distribución de citas (asistencias e inasistencias) por alcoholismo.....	18
Figura 14. Distribución de citas (asistencias e inasistencias) por grado de discapacidad	18
Figura 15. Distribución de citas (asistencias e inasistencias) por discapacidad	19
Figura 16. Distribución de citas (asistencias e inasistencias) según SMS recibido	19
Figura 17. Mapa de distribución de citas médicas por barrio y centros médicos (KMeans)	21
Figura 18. Distribución de citas (asistencias e inasistencias) según el día programado	23
Figura 19. Paciente con fecha de programación posterior a la fecha de la cita	23
Figura 20. Modificación de los valores negativos de “Time_SchDay_to_AppDay” a 0.....	23
Figura 21. Asignación del valor -1 a pacientes sin citas previas en “Days_since_last_App”	23
Figura 22. Distribución de citas (asistencias e inasistencias) agrupadas por cluster	24
Figura 23. Eliminación de cluster con bajo número de citas médicas	24
Figura 24. Matriz de correlación sin reducción de dimensionalidad	25
Figura 25. Transformaciones del dataset	26
Figura 26. Estandarización con StandardScaler	27
Figura 27. Distribución de citas por edad sin estandarizar (izquierda) y estandarizado (derecha) ..	27
Figura 28. Dataset Entrenamiento - Distribución de citas (sin balancear, SMOTE-ENN, ADASYN)....	27
Figura 29. Aplicación de SMOTE-ENN.....	28
Figura 30. Aplicación de ADASYN.....	28
Figura 31. Aplicación de PCA.....	28
Figura 32. Resultado de aplicación de PCA	28
Figura 33. Determinación del tamaño del conjunto de prueba	30
Figura 34. Determinación de la proporción de no asistencia.....	30
Figura 35. Hiperparámetros en regresión logística.....	31
Figura 36. Hiperparámetros en árbol de decisión	31
Figura 37. Pérdida por época en red neural.....	32
Figura 38. Resultados datos de entrenamiento por modelo para cada dataset.....	34
Figura 39. Resultados datos de prueba por modelo para cada dataset.	34
Figura 40. Marco de decisiones de un Sistema SAS.....	38
Figura 41. Reglas Asignación de Citas para los Escenarios 1 y 2.	40
Figura 42. Ejemplo de caso de Escenario 3 con Probabilidad de Show.	41
Figura 43. Fechas descartadas del Set de Pruebas	43
Figura 44. Coste Escenario 1 - Tradicional RBA sin Overbooking.....	45
Figura 45. Costes Escenario 2 - Tradicional RBA con Overbooking.....	45
Figura 46. Interpolación Polinómica en la forma de Lagrange.....	46

Figura 47. Número Pacientes por Consulta para un caso particular del Escenario 3	48
Figura 48. Costes Escenario 3 - Función lineal del “Listón ProbShow”	49
Figura 49. Costes Escenario 3 - Función cuadrática del “Listón ProbShow”	49
Figura 50. Mejores Costes del Sistema SAS por Escenario	50
Figura 51. Ahorros Económicos Anuales para Centro médico con 20 consultas médicas	51
Figura 52. Diagrama asistente virtual	52
Figura 53. Asistente virtual – Configuración de OpenAI.....	53
Figura 54. Asistente virtual - Interacción inicial	54
Figura 55. Asistente virtual – Esquema JSON de la función <i>iniciar_agendar_cita</i>	56
Figura 56. Asistente virtual – Autenticación Google Sheets.....	57
Figura 57. Asistente virtual - Base de datos citas médicas (Excel)	57
Figura 58. Asistente virtual - Agendar cita médica	58
Figura 59. Asistente virtual - Consultar citas médicas agendadas	58
Figura 60. Asistente virtual - Cancelación de cita médica	59
Figura 61. Asistente virtual - Orientación sobre especialidades	59
Figura 62. Asistente virtual - Información sobre medicamentos	60

Anexos

Anexo 1 – Cronograma del proyecto

El siguiente cronograma detalla las fechas de inicio y término de cada entrega, junto con el contenido relevante para cada una. Además, se describe la dinámica de trabajo y las reuniones semanales para asegurar el seguimiento y la coordinación del proyecto.

Reuniones semanales

- **Frecuencia:** Todos los viernes
- **Duración:** 2 horas
- **Objetivo:** Revisar el avance semanal, resolver dudas y organizar la distribución del trabajo.
- **Dinámica:** Cada miembro del equipo presenta su progreso y se discuten aspectos relevantes del proyecto. Se asignan tareas para la semana siguiente.

Proceso de trabajo

- **Desarrollo del proyecto:** Todas las personas del equipo (4 miembros) participan en el desarrollo del proyecto.
- **Distribución del contenido del documento:** Para la elaboración del documento final, los apartados se distribuyen de forma individual. Cada miembro se encarga de redactar su sección correspondiente.
- **Revisión conjunta:** Posteriormente, se realiza una revisión conjunta para integrar y unificar el contenido del documento.

Cronograma y entregas

Entrega	Fecha de inicio	Fecha de término	Contenido relevante
Entrega 1	23 Feb 2024	9 Abr 2024	Planteamiento del problema, posibles soluciones, análisis preliminar de bases de datos.
Entrega 2	19 May 2024	4 Jun 2024	Modelos de predicción de non show, algoritmo para optimización de agendas médicas, planteamiento de modelo de chatbot.
Entrega 3	5 Jun 2024	2 Ago 2024	Desarrollo final de modelos de predicción, optimización de algoritmos, desarrollo preliminar del Asistente Virtual.
Entrega final preliminar	19 Ago 2024	20 Sep 2024	Documento final del TFM, unificación de trabajo, conclusiones finales, Resumen Ejecutivo.
Entrega final	21 de Sep 2024	1 de Oct 2024	Entrega final del proyecto.

A continuación, se detalla cada entrega para ofrecer información más específica sobre el trabajo realizado en cada una.

Entrega 1

Fecha de inicio: 23 Feb 2024

Fecha de término: 9 Abr 2024

Contenido:

1. Búsqueda de base de datos de no show de pacientes
 - a. Planteamiento del problema y posibles soluciones:
 - i. Identificar y analizar las causas de no asistencia a citas médicas.
 - ii. Investigar posibles soluciones para mejorar la asistencia.
 - b. Análisis preliminar de las bases de datos:
 - i. Recopilar y analizar datos sobre no asistencia a citas médicas.
 - ii. Identificar tendencias y patrones en los datos.

Entrega 2

Fecha de inicio: 19 May 2024

Fecha de término: 4 Jun 2024

Contenido:

1. Desarrollo de modelos de predicción de no show (Fase 1):
 - a. Finalización del análisis exploratorio de datos (EDA): Analizar y visualizar los datos para comprender mejor su estructura y distribución.
 - b. Planteamiento de hipótesis con particionamiento del dataset: Dividir el conjunto de datos en varios subconjuntos, cada uno representando una hipótesis subyacente sobre la predicción de asistencias a citas médicas.
 - c. Optimización de modelos de machine learning para predicción de inasistencia de pacientes:
 - i. Búsqueda de mejores resultados de particiones: Evaluar y comparar diferentes particiones del dataset.
 - ii. Búsqueda de mejores modelos de machine learning y mejor red neuronal: Evaluar y comparar diferentes modelos de machine learning y redes neuronales.
 - iii. Optimización de hiperparámetros de modelos de machine learning: Ajustar los parámetros de los modelos para mejorar su rendimiento.
 - iv. Conclusiones iniciales comparando nuestros resultados con las bibliografías revisadas: Comparar los resultados obtenidos con los reportados en la literatura.
2. Planteamiento de algoritmo para optimización de agendas médicas según resultados de fase 1 (Fase 2):
 - a. Finalizar búsqueda bibliográfica sobre el tema: Investigar y recopilar información sobre algoritmos de optimización de agendas médicas.
 - b. Selección de hipótesis y función de coste. Determinar las principales hipótesis que delimiten el problema a resolver, así como la función de coste a calcular.
3. Planteamiento de modelo de asistente virtual de interacción con pacientes (Fase 3):
 - a. Iniciar búsqueda bibliográfica sobre el tema: Investigar y recopilar información sobre asistente virtual médicos.
 - b. Definir el framework con el que elaborar el asistente virtual: Seleccionar un framework para desarrollar el asistente virtual.
 - c. Definir modalidad de interacción del asistente virtual.

Entrega 3

Fecha de inicio: 4 Jun 2024

Fecha de término: 2 Ago 2024

Contenido:

1. Desarrollo final de modelos de predicción de no show (Fase 1):

- a. Optimización de modelos de Machine y Deep Learning para cada grupo de usuarios: Ajustar los modelos para cada grupo de usuarios.
 - b. Conclusiones de nuestros resultados con las bibliografías revisadas: Comparar los resultados obtenidos con los reportados en la literatura.
2. Algoritmos para optimización de agendas médicas según resultados de fase 1 (Fase 2):
 - a. Terminar de definir la función de coste a utilizar: Implementar un coste exponencial a los tiempos de espera del paciente.
 - b. Analizar Reglas de Asignación de Cita tradicionales: Estudiar, comparar y calcular el coste de los algoritmos utilizando estas reglas que tratan de optimizar agendas médicas, con especial interés en aquellas que generan overbooking.
 - c. Definir y evaluar una función de asignación de cita que utilice las probabilidades de asistencia a la cita médica obtenidas en la Fase 1.
 - d. Desarrollar un algoritmo de optimización: necesario para optimizar la función de asignación de cita mencionada en el punto anterior, minimizando los costos de no asistencia y overbooking.
 - e. Comparar costes de algoritmos: Evaluar y comparar los resultados de los algoritmos.
 - f. Generar conclusiones: Presentar los resultados y conclusiones sobre la optimización de agendas médicas.
3. Desarrollo preliminar de Asistente Virtual de interacción con pacientes (Fase 3):
 - a. Crear/inventar bases de datos: Crear bases de datos de médicos por especialidad y ubicación, y agendas médicas con información de médico, especialidad, ubicación y fecha (libre o Id. Paciente).
 - b. Configuración del Asistente Virtual de 3^a generación: Redactar Instrucciones y definir archivos para usar en el file search, así como los parámetros de Temperatura y Top-p (nucleus sampling) para controlar los flujos de conversación entre el Asistente y los Usuarios, permitiendo proporcionar información básica sobre síntomas y medicamentos.
 - c. Crear funciones de búsqueda, lectura y escritura a dichas bases de datos: Implementar funciones para interactuar con las bases de datos
 - d. Probar y refinar el Asistente Virtual.
 - e. Generar conclusiones: Presentar los resultados y conclusiones sobre el Asistente Virtual.
4. Se realiza un planteamiento teórico de una integración de los 3 puntos anteriores (Fase 4).

Entrega final preliminar

Fecha de inicio: 19 Ago 2024

Fecha de término: 20 Sep 2024

Contenido:

1. Elaboración y compilación del documento final del TFM:
 - a. Unificación de trabajo de las 3 fases: Integrar los resultados de las tres fases.
 - b. Conclusiones finales.
 - c. Revisión final y cita de bibliografías.

Entrega final

Fecha inicio: 21 Sep 2024

Fecha de término: 1 Oct 2024

Contenido:

1. Desarrollo de presentación:
 - a. Crear una presentación en PowerPoint que resuma los hallazgos y conclusiones del proyecto.

- b. Incluir gráficos, visualizaciones y otros elementos para comunicar de manera efectiva los resultados.
- c. Revisar la presentación para su entrega final.

Anexo 2 – Estimación de recursos económicos

Fase 1: Predicción de Asistencia a Citas Médicas

Costos asociados:

- **RRHH:** Esta fase, pensada para desarrollarse en 6 meses, requiere un equipo especializado formado por 1 analista o ingeniero del dato, para realizar el EDA y el Feature Engineering, y 1 ingeniero de IA para desarrollar los distintos entrenamientos y escoger el mejor modelo predictivo. **Se considera necesario mínimo 4 meses para el EDA y FE, y otros 2 meses para el diseño, entrenamiento y evaluación de los modelos de IA.**
 - RRHH: $50.000 \text{ €/año} \times 1 \text{ analista/desarrollador} \times 0,5 \text{ años} = 25,000 \text{ €}$
- **Infraestructura (PaaS de Google Cloud Platform):** 1 GPU para entrenamiento de los modelos de DP (durante los últimos 2 meses que dura el entrenamiento de estos algoritmos), 2 CPU's para el trabajo inicial con los datos y el entrenamiento de los modelos de ML, así como discos duros para el almacenamiento de los datos.
 - 1 GPU NVIDIA V100 de 16GB de memoria: $2,81 \text{ €/h} \times 360\text{h/mes} \times 2 \text{ meses} = 2.023 \text{ €}$
 - 2 CPU's con 6,5 GB de memoria cada una (N1-highmem-2): $105 \text{ €/mes} \times 6 \text{ meses} = 210 \text{ €}$
 - 1 unidad local SSD de 375 GiB: $0,1 \text{ €/mesGB} \times 6 \text{ meses} \times 400 \text{ GiB} = 240 \text{ €}$
- **Licencias y software:** Herramientas para desarrollo y gestión de datos (Python, TensorFlow, MySQL) son opensource, o ya vienen incluidas en la plataforma ofrecida la PaaS de GCP.
- **Total estimado para Fase 1: 27.473 €**

Fase 2: Implementación de un Sistema de Overbooking

Costos asociados:

- **RRHH:** Se necesitará 1 desarrollador adicional, conocedor del negocio, para crear las reglas del sistema de overbooking y definir la fórmula de coste. **Se considera necesario mínimo por 3 meses.**
 - RRHH: $50.000 \text{ €/año} \times 1 \text{ desarrollador} \times 0,25 \text{ años} = 12.500 \text{ €}$
- **Infraestructura (PaaS de Google Cloud Platform):** Sirve la misma utilizada en la Fase 1, pero sin la necesidad de la GPU.
 - Servidor de desarrollo y almacenamiento de datos: $75 \text{ €/mes} \times 3 \text{ meses} = 225 \text{ €}$
- **Total estimado para Fase 2: 12.725 €**

Fase 3: Creación de un Asistente Virtual Basado en NLP

Costos asociados:

- **RRHH:** El desarrollo del asistente virtual lo liderará un ingeniero especialista en Procesamiento de Lenguaje Natural (NLP) y en Prompt Engineering, el cual se encargará de modelar las instrucciones y funciones del Asistente, pero se requiere otro ingeniero de desarrollo para la integración de las APIs y los varios canales de distribución (Telegram, WhatsApp, Messenger, etc), así como su mantenimiento en el servidor. **Se considera una duración de esta fase mínima de 3 meses.**
 - Especialista IA: $65.000 \text{ €/año} \times 0,25 \text{ años} = 16.250 \text{ €}$
 - Desarrollador de software: $35.000 \text{ €/año} \times 0,25 \text{ años} = 8.750 \text{ €}$
- **Infraestructura (PaaS de Google Cloud Platform):** Para alojar y ejecutar el asistente.
 - Servidor de producción: $75 \text{ €/mes} \times 6 \text{ meses} = 450 \text{ €}$
- **Licencias de software:** Para las herramientas de desarrollo NLP.

- Licencias de software: $500 \text{ €/año} \times 2 \text{ usuarios} \times 0.25 \text{ años} = 250 \text{ €}$
- **Uso de los modelos de NLP (Chat GPT-4o):** Durante la fase de pruebas del Asistente Virtual.
 - Platform OpenAI: $400 \text{ €/mes} \times 3 \text{ meses} = 1,200 \text{ €}$
- **Total estimado para Fase 3:** 26.900 €

Fase 4: Integración y Mejora Continua del Sistema

Costos asociados:

- **RRHH:** Se considera un gerente del Proyecto, especialista en IA y conocedor del Negocio, que se encargue de supervisar y atender la mejora continua del Sistema; un analista de datos para mantener la infraestructura del dato necesaria, así como desarrollar un análisis continuo de los datos que se van integrando al modelo de predicción, y 1 desarrollador de software para mantener la integración del sistema y del asistente. Los 2 primero sólo dedicando un cuarto de jornada.
 - Gerente del Proyecto: $60.000 \text{ €/año} \times 0,25 = 15.000 \text{ €/año}$
 - Analista de Datos: $45.000 \text{ €/año} \times 0,25 = 11.250 \text{ €/año}$
 - Desarrollador de Software: $35.000 \text{ €/año} \times 1 = 35.000 \text{ €/año}$
- **Infraestructura (PaaS de Google Cloud Platform):** Se considera la necesidad a medio plazo de duplicar el servidor de producción (otras 2 CPU's iguales y 750 GiB de almacenamiento adicionales) para la integración del sistema, así como otras 360 horas al año para actualizar el modelo de predicción (un mes al año).
 - 1 GPU NVIDIA V100 de 16GB de memoria: $2,81 \text{ €/h} \times 360\text{h/mes} \times 1 \text{ mes} = 1.012 \text{ €/año}$
 - 4 CPU's con 6,5 GB de memoria cada una (N1-highmem-4): $210 \text{ €/mes} \times 12 \text{ meses} = 2.520 \text{ €/año}$
 - 2 unidades locales SSD de 375 GiB: $2 \times 0,1 \text{ €/mes} \times 12 \text{ meses} \times 400 \text{ GiB} = 960 \text{ €/año}$
- **Licencias de software:** Para las herramientas de desarrollo NLP.
 - Licencias de software: $500 \text{ €/año} \times 3 \text{ usuarios} = 1.500 \text{ €/año}$
- **Uso de los modelos de NLP (Chat GPT-4o):** Durante la fase de mantenimiento del Asistente Virtual.
 - Platform OpenAI: $500 \text{ €/mes} \times 12 \text{ meses} = 6.000 \text{ €/año}$

- **Total estimado para Fase 4:** 73.242 €/año

Resumen total de costos aproximado:

- **Fase 1:** 27.500 €
- **Fase 2:** 13.000 €
- **Fase 3:** 27.000 €
- **Fase 4:** 75.000 €/año
- **TOTAL global estimado y aproximado:**
 - Desarrollo: 70.000 €
 - Mantenimiento: 75.000 €/año

Anexo 3 – Prompt asistente

Instrucciones especificadas en la configuración del asistente en la plataforma OpenAI.

“Bienvenido, Basilio, eres el asistente virtual del Hospital Clínico, diseñado para ayudar a los usuarios de diferentes formas y en cualquier idioma.

Identidad y Origen

Te llamas Basilio en honor a San Basilio el Grande, quien fundó el primer hospital para brindar asistencia médica en Cesárea de Capadocia (actualmente Kayseri, Turquía). No dudes en compartir esta

información si te preguntan por el origen de tu nombre. Guardarrail: No ofrecer esta información sin que el usuario lo solicite explícitamente.

Primera Interacción

En la primera interacción con el usuario, sin informar al usuario, iniciarás usando el code interpreter, para calcular la fecha actual, la cual la mantendrás almacenada para usar posteriormente. Una vez la tengas almacenada, te presentarás, sin ser repetitivo y siendo amigable y gracioso, especificando la fecha que has obtenido con el code interpreter y un resumen de las funcionalidades especificadas en "Misión principal". Guardarrail: No almacenar ni procesar información personal del usuario en esta etapa, salvo el mínimo necesario para la interacción. Guardarrail: No repetir este proceso en interacciones posteriores, y no ofrecer funcionalidades no solicitadas explícitamente.

Interacciones

En las interacciones con el usuario debes tener en cuenta que si su consulta está relacionada con: citas médicas, medicamentos, información sobre el hospital, especialidades médicas, horario de apertura, ubicación de sedes, cancelación de citas, modificación de citas, agenda de nuevas citas, puedes proporcionar una respuesta detallada; si no lo está, da una respuesta general, breve y educada indicando que el tema no está cubierto por el Hospital Clínico. Por ejemplo: "Lo siento, no puedo ayudar con eso" para temas que sea de deportes, viajes, chistes, historia de cualquier ámbito, aparatos. En cualquier a del os casos se amigable y cercano al paciente.

Información sobre Sedes

La información sobre las sedes del Hospital Clínico será extraída del documento adjunto, usando la herramienta File Search. Guardarrail: No adivinar ni inventar ubicaciones si no están en el documento. Si te piden un mapa, usa el Code Interpreter para generar la ubicación exacta. Guardarrail: Verificar la precisión de la información antes de entregarla y no ofrecer mapas de ubicaciones no relacionadas con el Hospital Clínico.

Horario de Apertura

El Hospital Clínico está abierto de lunes a viernes de 8:00 a 19:00. Guardarrail: No proporcionar información incorrecta sobre horarios, ni sugerir atención fuera de los horarios establecidos. La última cita está disponible a las 18:40 y las citas se pueden agendar en intervalos de 20 minutos. Guardarrail: No permitir citas fuera de este rango de horarios.

Especialidades Médicas

Las especialidades disponibles en el Hospital Clínico son:

- Medicina General
- Pediatría
- Dermatología
- Cardiología
- Ginecología

Guardarrail: No sugerir especialidades médicas no listadas, ni inventar información sobre especialidades no disponibles en el hospital.

Misión Principal

Tu principal misión es:

1. Identificación de Especialistas: Ayuda a los usuarios a identificar con qué especialidad debe consultar, basándote en los síntomas descritos. Guardarrail: No proporcionar diagnósticos médicos ni suponer condiciones de salud serias. Siempre ofrecer orientación para consultar con un profesional adecuado.

2. Información sobre Citas Médicas: Proporciona información sobre citas médicas ya agendadas, como la fecha y hora. Guardarrail: No acceder ni compartir información de citas médicas sin la solicitud explícita del usuario. Asegurarse de cumplir con las políticas de privacidad del hospital.

3. Cancelar o Modificar Citas: Cancela o modifica una cita ya agendada. Si no hay disponibilidad en la fecha solicitada, pregunta si el usuario desea ver la siguiente fecha disponible o especificar una nueva fecha. Guardarrail: No modificar ni cancelar citas sin confirmación explícita del usuario.

4. Agendar Nuevas Citas Médicas: Agendar citas solicitando la especialidad médica o doctor concreto, si es necesario. Guardarrail: No permitir la programación de citas fuera de los horarios establecidos del hospital, ni agendar citas sin recibir información suficiente sobre la especialidad o médico deseado.

5. Información sobre Medicamentos. Proporciona información sobre medicamentos usando la base de datos de tu conocimiento, subrayando la obligatoriedad de la prescripción médica cuando sea necesario. Si te solicitan información sobre un medicamento que no conoces, sugiere consultar con un profesional del hospital. Ofrece información sobre el propósito y las instrucciones de uso de los medicamentos, y si es necesario, indica el doctor más adecuado para una consulta y pregunta si desean agendar una cita. Guardarrail: No recomendar medicamentos sin una consulta médica previa. Evita sugerir tratamientos sin la orientación de un profesional.

Reglas de Consulta y Diagnóstico

Evita dar diagnósticos médicos. Guardarrail: Nunca ofrecer respuestas que puedan ser interpretadas como diagnósticos o instrucciones de tratamiento. Proporciona orientación sobre los especialistas adecuados según los síntomas descritos. Guardarrail: Si la consulta es sobre un síntoma grave, sugerir siempre que se consulte directamente con un médico.

Tono y Comunicación

Mantén un tono profesional, empático y respetuoso en todas las interacciones, asegurándote de ser amigable y cercano al paciente. Utiliza un lenguaje accesible y cálido que fomente la confianza y la comunicación abierta. Guardarrail: Evitar tonos sarcásticos, bromas o comentarios que puedan resultar ofensivos. Solo solicitar información adicional cuando sea estrictamente necesario para completar una acción. Guardarrail: Si el usuario está molesto o realiza preguntas inapropiadas, redirigir la conversación de manera calmada hacia un tema relevante de salud o atención médica. Guardarrail para solicitud de chistes: Si te piden contar un chiste, declina amablemente y redirige hacia temas relacionados con el hospital. No contar chistes en ningún contexto."