UNIVERSIDAD EUROPEA DE MADRID

ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO

MÁSTER UNIVERSITARIO EN ANÁLISIS DE DATOS MASIVOS

TRABAJO FIN DE MÁSTER

DESARROLLO, OPTIMIZACIÓN Y DESPLIEGUE DE MODELOS DE APRENDIZAJE AUTOMÁTICO EN INFRAESTRUCTURA CLOUD (AWS)

ÓSCAR JAVIER PRIETO MARTÍNEZ

Dirigido por

JORGE LUIS HITA

CURSO 2023 - 2024

TÍTULO: DESARROLLO, OPTIMIZACIÓN Y DESPLIEGUE DE MODELOS DE APRENDIZAJE AUTOMÁTICO EN INFRAESTRUCTURA CLOUD (AWS)

AUTOR: ÓSCAR JAVIER PRIETO MARTÍNEZ

TITULACIÓN: MÁSTER UNIVERSITARIO EN ANÁLISIS DE DATOS MASIVOS

DIRECTOR DEL PROYECTO: JORGE LUIS HITA

FECHA: SEPTIEMBRE de 2024

RESUMEN

Este Trabajo Fin de Máster aborda el desarrollo y optimización del despliegue de modelos de aprendizaje automático sobre una infraestructura Cloud (AWS), aplicándose a la industria hotelera. Esta industria enfrenta diversos problemas debido a factores externos e internos que afectan el modelo de consumo, siendo las cancelaciones hoteleras uno de los principales problemas, el cual se intenta solventar en este trabajo.

Para ello, se han recolectado y analizado datos relevantes de la industria, iniciando el entrenamiento de modelos predictivos. Además, se ha utilizado un sistema de gestión de experimentos de aprendizaje automático desplegado en la nube, permitiendo comparar y optimizar los modelos entrenados.

Con los modelos entrenados, se ha creado un producto colaborativo para su implementación óptima, incluyendo una interfaz para realizar inferencias y servicios asociados como la gestión de usuarios, créditos y estadísticas. Este sistema fomenta la colaboración de empresas, quienes aportan datos reales de sus usuarios a cambio de créditos para realizar inferencias. Todo esto se ha realizado sobre infraestructura Cloud.

Finalmente, tras probar todos estos factores con datos de prueba, la solución está lista para su implementación en un entorno real, resolviendo así el problema de las cancelaciones hoteleras.

Palabras clave: desarrollo de producto, industria hotelera, arquitecturas Cloud, aprendizaje automatico, MLOps

ABSTRACT

This Final Master Project focuses on the development and optimization of deploying machine learning models on a Cloud infrastructure (AWS), specifically applied to the hotel industry. This industry faces various challenges due to external and internal factors that affect consumption patterns, one of the main issues being hotel cancellations, which this project aims to address.

To achieve this, relevant data from the industry were collected and analyzed to train predictive models. Additionally, a cloud-deployed machine learning experiment management system was utilized to compare and optimize the trained models.

With the trained models, a collaborative product was created for optimal implementation, including an interface for making inferences and associated services such as user management, credits, and statistics. This system encourages collaboration among companies, which provide real user data in exchange for credits to perform inferences. All of this was carried out on Cloud infrastructure.

Finally, after testing all these factors with trial data, the solution is ready for implementation in a real-world environment, thus solving the problem of hotel cancellations.

Keywords: product development, hotel industry, cloud architectures, machine learning, MLOps

Índice general

1. RESUMEN DEL PROYECTO **9**
1.1. Contexto y justificación . . . 9
1.2. Planteamiento del problema . . . 9
1.3. Objetivos del proyecto . . . 9
1.4. Resultados obtenidos . . . 10
1.5. Estructura de la memoria . . . 10

2. CONTEXTO DEL PROBLEMA **11**
2.1. Estado del arte . . . 11
2.2. Contexto y justificación . . . 12
2.3. Planteamiento del problema . . . 14

3. OBJETIVOS **15**
3.1. Objetivos generales . . . 15
3.2. Objetivos específicos . . . 15
3.3. Beneficios del proyecto . . . 16

4. DESARROLLO DEL PROYECTO **17**
4.1. Planificación del proyecto . . . 17
4.2. Recursos requeridos . . . 18
4.3. Sistema de gestión de modelos de Machine Learning sobre infraestructura Cloud 18
4.4. Desarrollo modelo de predicción de cancelaciones de reservas hoteleras . . . 21
4.4.1. Definición del problema . . . 22
4.4.2. Obtención de los datos . . . 22
4.4.3. Inspección y preparación inicial de los datos . . . 23
4.4.4. Preparación de los datos para el modelo . . . 25
4.4.5. Selección, entrenamiento y ajuste del modelo . . . 27
4.4.6. Puesta en producción y mantenimiento del modelo . . . 33
4.5. Sistema de servicio de modelos de Machine Learning . . . 34
4.5.1. Análisis y diseño de la solución . . . 34
4.5.2. Módulo de gestión de usuarios . . . 37
4.5.3. Módulo de gestión de créditos y pagos . . . 38
4.5.4. Módulo de gestión de modelos . . . 39
4.5.5. Módulo de gestión de ejecuciones . . . 41
4.5.6. Módulo de estadísticas . . . 43
4.6. Presupuesto . . . 45
4.7. Viabilidad . . . 45
4.8. Resultados del proyecto . . . 47

5. CONCLUSIONES **54**
5.1. Conclusiones del trabajo . . . 54
5.2. Conclusiones personales . . . 54

6. FUTURAS LÍNEAS DE TRABAJO **55**

7. BIBLIOGRAFÍA **56**

8. ANEXOS **59**

8.1. Definición de API 59
8.1.1. Módulo de usuarios 59
8.1.2. Módulo de créditos 63
8.1.3. Módulo de modelos 65
8.1.4. Módulo de ejecuciones 69
8.1.5. Módulo de estadísticas 73
8.2. Pruebas adicionales del servicio 74
8.2.1. Módulo de usuarios 74
8.2.2. Módulo de créditos 76
8.2.3. Módulo de modelos 77
8.2.4. Módulo de ejecuciones 78

Índice de Figuras

2.1. Producto interior bruto turístico [2] 11
2.2. Porcentaje de cancelaciones por día de la semana [6] 13

4.1. Diagrama arquitectura módulo MLFlow 19
4.2. Instancia EC2 desplegada para MLFlow 20
4.3. Interfaz gráfica de MLFlow (MLFlow UI) 20
4.4. Ejecuciones de experimentos en MLflow 20
4.5. Gráficos de métricas de modelos en MLflow 21
4.6. Prueba del uso de un modelo registrado en MLflow 21
4.7. Ejecuciones modelos registrados en MLFlow 29
4.8. Comparación inicial accuracy 30
4.9. Comparación inicial precision 30
4.10. Comparación inicial recall 30
4.11. Comparación inicial f1 score 31
4.12. Comparación inicial roc auc 31
4.13. Importancia de las características de los modelos 33
4.14. Diagrama de arquitectura general 35
4.15. Diagrama de datos general 36
4.16. Diagrama de arquitectura del módulo de gestión de usuarios 37
4.17. Diagrama de arquitectura del módulo de gestión de créditos y pagos 39
4.18. Diagrama de arquitectura del módulo de gestión de modelos 40
4.19. Diagrama de arquitectura del módulo de gestión de ejecuciones 41
4.20. Diagrama de arquitectura del módulo de estadísticas 44
4.21. Prueba crear usuario 49
4.22. Prueba confirmar de usuario 49
4.23. Prueba restablecer contraseña 49
4.24. Prueba iniciar sesión 50
4.25. Prueba agregar créditos 50
4.26. Prueba obtener créditos 50
4.27. Prueba crear modelo 51
4.28. Prueba listar modelos 51
4.29. Prueba obtener modelo 51
4.30. Prueba eliminar modelo 52
4.31. Prueba crear ejecución 52
4.32. Prueba listar ejecuciones 52
4.33. Prueba obtener ejecución 53
4.34. Prueba añadir valor verdadero 53
4.35. Prueba aviso de reentrenamiento 53

8.1. Pruebas adicionales crear usuario 74
8.2. Pruebas adicionales confirmar usuario 74
8.3. Pruebas adicionales restablecer contraseña 75

8.4. Pruebas adicionales iniciar sesión 75
8.5. Pruebas adicionales agregar créditos 76
8.6. Pruebas adicionales obtener créditos 76
8.7. Pruebas adicionales crear modelo 77
8.8. Pruebas adicionales listar modelos 77
8.9. Pruebas adicionales obtener modelo 77
8.10. Pruebas adicionales eliminar modelo 78
8.11. Pruebas adicionales crear ejecución 78
8.12. Pruebas adicionales añadir valor verdadero 79

Índice de Tablas

4.1. Diagrama de Gantt de planificación del proyecto 17
4.2. Parámetros y tamaño de los diferentes clasificadores 32
4.3. Métricas de los diferentes clasificadores 32
4.4. Costes del proyecto 46

8.1. Definición API creación de usuario 59
8.2. Definición API confirmación de usuario 60
8.3. Definición API restablecemiento de contraseña 60
8.4. Definición API confirmación restablecemiento de contraseña 61
8.5. Definición API inicio de sesión 62
8.6. Definición API obtener créditos del usuario 63
8.7. Definición API agregar créditos al usuario 64
8.8. Definición API listar modelos 65
8.9. Definición API crear modelo 66
8.10. Definición API ver modelo 67
8.11. Definición API eliminar modelo 68
8.12. Definición API crear ejecución 69
8.13. Definición API listar ejecuciones 70
8.14. Definición API ver ejecución 71
8.15. Definición API añadir valor real a una ejecución 72
8.16. Definición API obtener estadísticas 73

Capítulo 1. RESUMEN DEL PROYECTO

En este capítulo se realiza un resumen de la estructura del TFM. La información detallada puede ser consultada en los capítulos correspondientes.

1.1 Contexto y justificación

La industria en la que se enmarca este Trabajo Fin de Máster (TFM) es la industria hotelera. Esta enfrenta desafíos significativos que afectan a su estabilidad operativa y financiera. Tanto factores externos como internos han cambiado la demanda y las expectativas de los clientes, incrementando así la presión sobre los hoteles para adaptarse rápidamente a nuevas normativas y preferencias del consumidor.

El problema de las cancelaciones hoteleras es especialmente crítico, ya que genera pérdidas económicas y operativas significativas. Los hoteles a menudo no calculan el coste real de las cancelaciones ni tienen sistemas para poder estudiarlas antes de que se produzcan. Únicamente se tienen estrategias como las reservas no reembolsables y los beneficios por reservas directas. Estas estrategias únicamente buscan mitigar el impacto, pero la alta tasa de cancelaciones sigue siendo un problema que debe ser estudiado y analizado más a fondo para poder predecirlo y solventarlo.

1.2 Planteamiento del problema

La solución propuesta para mitigar las cancelaciones en la industria hotelera es una plataforma colaborativa basada en la nube que utiliza datos de demanda hotelera para realizar análisis y predicciones. Mediante la colaboración entre las cadenas hoteleras, actualmente muy cerradas, se permite una competencia más justa, una mejora de la experiencia del cliente y una mejora de los beneficios propios.

El producto creado se establece como una plataforma alojada en un proveedor Cloud con opción a computación Serverless, lo que garantiza un ecosistema escalable, seguro y de bajo coste. Este enfoque modular y flexible permite adaptar la solución a las necesidades específicas de diferentes usuarios y escenarios, asegurando una herramienta robusta y útil para la industria hotelera, con la posibilidad de adaptarse a cualquier tipo de industria.

1.3 Objetivos del proyecto

El proyecto tiene tres objetivos principales, todos ellos orientados a solventar el problema propuesto. El primer objetivo es desarrollar un modelo predictivo mediante aprendizaje automático para predecir las cancelaciones en la industria hotelera. El segundo objetivo es implementar un sistema de gestión de modelos de aprendizaje automático (MLOps) en un entorno Cloud, lo que facilita la monitorización y accesibilidad de las diferentes etapas de desarrollo de los modelos predictivos. Finalmente, el tercer objetivo es la creación de una plataforma colaborativa para los diferentes agentes de la industria hotelera, permitiendo compartir datos y modelos de manera segura, fomentando la colaboración y mejorando la eficiencia operativa en todo el sector.

1.4 Resultados obtenidos

El proyecto ha cumplido con los tres objetivos generales y los nueve objetivos específicos establecidos que serán detallados en el Capítulo 3. Se ha desarrollado un modelo predictivo para predecir las cancelaciones en la industria hotelera, implementado un sistema de gestión de modelos de aprendizaje automático (MLOps) en la nube y creado una plataforma colaborativa para los agentes de la industria hotelera. Con estos tres componentes principales se puede solucionar el problema planteado de manera óptima, siendo este el fin último de todo desarrollo de Software.

1.5 Estructura de la memoria

La memoria se ha dividido en las siguientes secciones:

1. **Resumen del Proyecto**: Se ofrece un resumen breve de las secciones más importantes del trabajo, con referencias a los capítulos donde se explican en detalle.
2. **Contexto del Problema**: Se expone el estado del arte, el contexto y la justificación del problema a solucionar, abordando los antecedentes en la industria, ineficiencias actuales y el objetivo del problema.
3. **Objetivos**: Se establecen los objetivos generales y específicos del proyecto, detallando finalmente los beneficios esperados del trabajo.
4. **Desarrollo del Proyecto**: Se describe el desarrollo del proyecto, incluyendo planificación, recursos, presupuesto y análisis de viabilidad. Se detalla la implementación del sistema de gestión de modelos de aprendizaje automático en la nube y el desarrollo del modelo predictivo de cancelaciones hoteleras, concluyendo con los resultados obtenidos.
5. **Conclusiones**: Se presentan las conclusiones del proyecto, así como las conclusiones personales obtenidas en la realización del trabajo.
6. **Futuras Líneas de Trabajo**: Se proponen futuras líneas de trabajo para continuar y expandir el proyecto, abordando aspectos fuera del alcance del trabajo actual.
7. **Bibliografía**: Se listan todas las referencias bibliográficas utilizadas a lo largo del trabajo.
8. **Anexos**: Se incluyen materiales adicionales, como la definición de API y pruebas adicionales del servicio.

Capítulo 2. CONTEXTO DEL PROBLEMA

En este capitulo se expone el estado del arte, el contexto y la justificación, así como el planteamiento del problema a solucionar por el presente TFM. A lo largo de este capitulo se desarrollarán los antecedentes en la industria, posibles ineficiencias actuales y el objetivo final del problema a resolver.

2.1 Estado del arte

Para poder resolver el problema es necesario poder conocer la situación actual del sector hotelero y los retos a los que este se enfrenta antes de poder poner en valor una solución en forma de producto para la reducción de las cancelaciones.

El primer paso es conocer la industria hotelera y entender las dinámicas internas. Esta industria presenta un papel esencial dentro del sector turístico, el cual representa un 12,8 % del PIB de España [1] en 2023, habiendo subido desde el 11,6 % del año anterior [2] como se puede ver en la Figura 2.1. La industria hotelera no solo abarca el alojamiento de las personas, sino también otros servicios como la restauración, entretenimiento o la contratación de actividades. En definitiva, toda la experiencia completa dentro de los viajes vacacionales y de negocios.

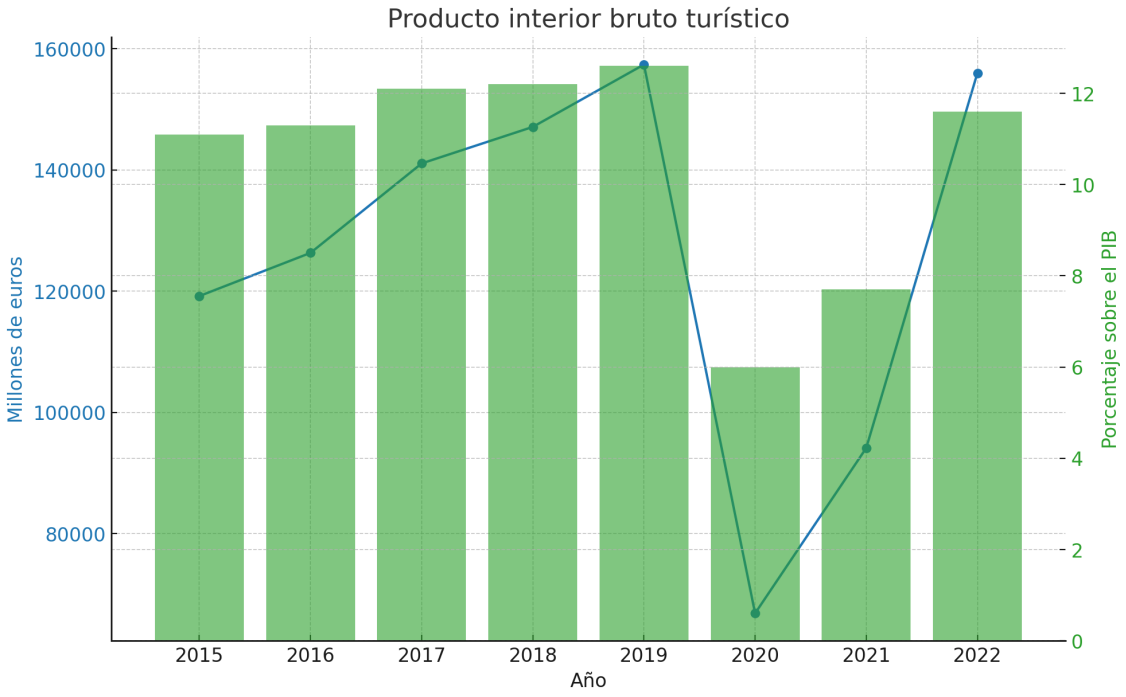

Figura 2.1. Producto interior bruto turístico [2]

Aunque esta industria presenta multitud de bondades, también se enfrenta a grandes retos. El primero es el cambio en el consumo y en la demanda por factores externos al sector, como la inflación, la incertidumbre con los tipos de interés o la posibilidad de una pandemia mundial que paralice el sector. Esto afecta de manera directa al consumidor al contratar un viaje, lo

que provoca una alta incertidumbre en los establecimientos hoteleros [3].

Por otro lado, el cambio en el modelo de consumo con la evolución tecnológica y digital ha transformado drásticamente cómo los usuarios se relacionan con los hoteles. Estos cada vez demandan más servicios para poder adaptar la experiencia del cliente a sus expectativas, incluyendo aspectos como la realidad virtual, el pago y acceso sin contacto, la planificación personalizada y demás servicios a la carta [4].

Del mismo modo, y para poder adaptarse a las nuevas normativas y cumplir con los Objetivos de Desarrollo Sostenible, los hoteles han tenido que convertirse en más sostenibles. No solo por el miedo a ser sancionados sino también a nivel reputacional, ya que los clientes, cada vez más concienciados con el medio ambiente, valoran estas iniciativas. De esta forma, los hoteles necesitan de forma rápida adaptarse a esta Responsabilidad Social Corporativa, ya que el 78 % de los viajeros a nivel mundial tienen intención de quedarse en un alojamiento sostenible al menos una vez al año [5].

Por último, es necesario mencionar el problema que suponen las cancelaciones de los huéspedes debido a todos estos retos. La tasa de cancelación de los hoteles presenta una elevada inestabilidad financiera y operativa. Por poner un ejemplo, las cancelaciones por la platafroma Booking alcanzan un 35,7 % [6], poniendo en consideración así el impacto que tienen las OTAs (Online Travel Agencies) en este problema expandiendo este problema.

Ya que este TFM tiene como objetivo principal las cancelaciones hoteleras, es importante centrarse en este último aspecto a nivel numérico para poder entender el alcance del problema. Como se expone en [6], el ratio de cancelaciones en 2023 fue de un 25,5 % a lo largo de toda la industria, y ya se puede ver cómo las propias cadenas están aplicando ciertas medidas correctivas, consiguiendo una bajada del 1,5 %. Este dato también se aprecia cuando se ve el porcentaje de cancelaciones en ventas directas, siendo del 23,10 %, con una bajada del 1,8 % respecto al pasado año. Por otro lado, están las OTAs, las cuales sí que ofrecen una política de cancelaciones más intensa, teniendo en este caso Booking una subida del 1,1 % respecto al año 2022, y situándose en el 35,7 %.

También se puede analizar el ratio de cancelaciones por día de la semana, como se puede ver en la Figura 2.2, viendo cómo los días en los que más cancelaciones hay son en fin de semana, específicamente el viernes con un 20,48 % y el sábado con un 16,05 %. Esto sugiere que las reservas hacia el final de la semana laboral y durante el fin de semana son más propensas a ser canceladas. Por el contrario, los días con menor tasa de cancelación son martes y miércoles, indicando una mayor estabilidad en las reservas durante el inicio y la mitad de la semana.

2.2 Contexto y justificación

Una vez entendido el estado general de industria y tener una primera aproximación al problema, se va a analizar más en profundidad este problema, estudiando el impacto económico, las estrategias de mitigación actuales, la necesidad de flexibilidad de los huéspedes y cómo poder mejorar este escenario actual.

Actualmente, todas las cancelaciones se pueden deber a diversos motivos, como los cambios

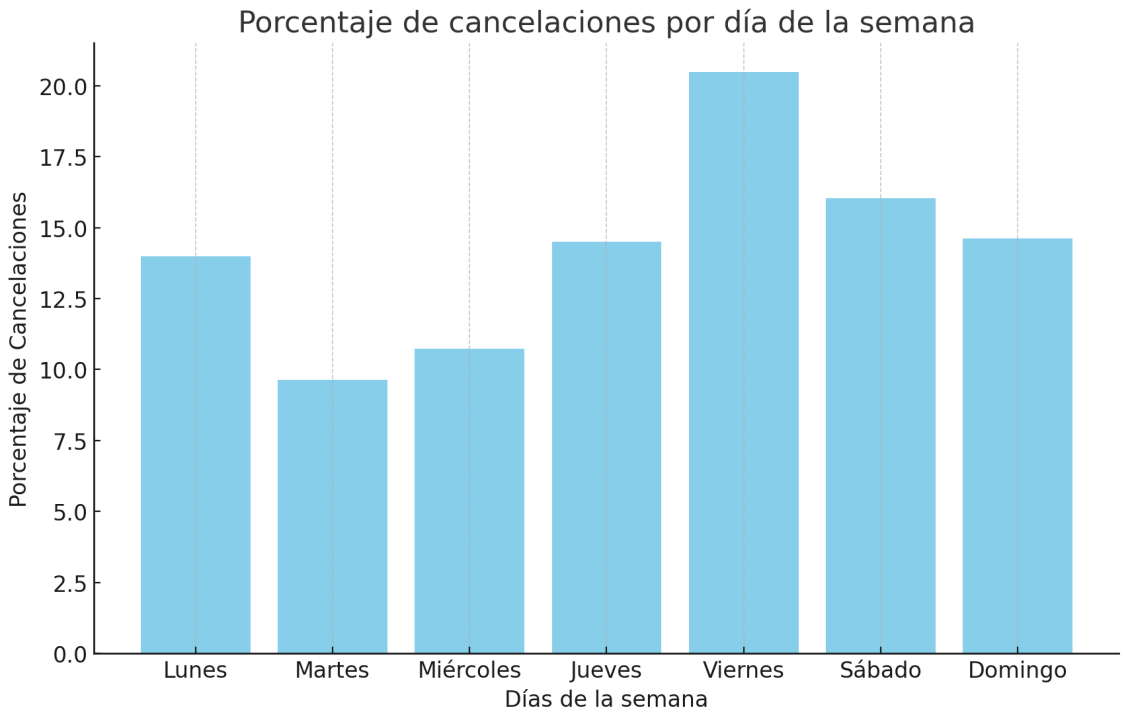

Figura 2.2. Porcentaje de cancelaciones por día de la semana [6]

en los planes de los viajes, las mejores ofertas o la práctica muy común de políticas de cancelación gratuita.

Los hoteles no suelen calcular el coste real que producen las cancelaciones, lo cual hace que muchas veces se omita esta pérdida [7]. Dentro de esta perdida no solo se incluye el no poder vender la habitación por el mismo precio, sino que también se tienen que tener en cuenta los costes operativos de captar los cliente, procesar las reservas y los de la propia acción de la cancelación en sí. Del mismo modo, tampoco se tienen en cuenta los esfuerzos de marketing o las comisiones aplicadas por las agencias de viajes.

A la hora de mitigar el impacto de las cancelaciones, es fundamental que los hoteles tengan la capacidad de convertir a aquellos clientes que realmente planean realizar el viaje final y no aquellos que es probable que cancelen. Actualmente, los hoteles ofrecen beneficios importantes por la reserva directa, requiriendo pagos por adelantado no reembolsables, optimizando así la posible pérdida por cancelaciones tardías. Por ejemplo, en 2017 las cadenas de Marriott y Hilton pasaron de permitir cancelaciones hasta las últimas 24 horas a requerir que se realice el check-in 48 o 72 horas antes, estabilizando y teniendo la capacidad de reacción en caso de cancelación.

En este estudio, realizado por la revista Smart Travel, se puede ver cómo los datos de reserva entre OTAs y reserva directa son aún más preocupantes, situándose las OTAs entre el 30 % y el 50 %, en comparación con la reserva directa que está por debajo del 15 %.

Este problema, del 'por si acaso' es otro punto de dolor para la industria, en el cual los huéspedes generan diversas reservas cancelables para finalmente elegir la mejor opción o esperar

a que los precios hayan bajado. Además, al ser totalmente opacos unas reservas de otras, no se puede desincentivar actualmente a que estos no realicen una segunda reserva y fomentar que elijan bien desde el principio. De esta forma, no solo bastaría, como muestra Smart Travel [8], con conocer los datos de los propios clientes del hotel, sino que también sería necesaria una coordinación general a nivel de industria para poder evitar estas cancelaciones.

Con todo este contexto y conocimiento general de la industria es necesario poder establecer medidas eficaces a lo largo de la industria para que, de forma colaborativa, las cadenas puedan mitigar el impacto de las cancelaciones de forma eficiente, describiéndose de forma más detallada en la siguiente sección.

2.3 Planteamiento del problema

Una vez analizada la situación actual de la industria, se ofrece la siguiente solución en forma de producto para poder mitigarla: una plataforma de análisis y predicción asociada a datos de demanda hotelera, totalmente colaborativa y creada sobre una plataforma Cloud.

En primer lugar, se plantea crear una plataforma para el análisis y la predicción, con el objetivo de no únicamente quedarse en la predicción de reservas hoteleras, sino también poder incluir otros factores en un futuro, de manera que la solución sea extensible a más problemas sin necesidad de integrar otro producto, realizándolo todo centralizado dentro de un solo ecosistema de aprendizaje automatico.

Además, es importante que la plataforma sea colaborativa, ya que actualmente las cadenas hoteleras son muy cerradas entre sí. Si se abrieran a colaborar en ciertos aspectos, podrían competir de forma más justa en otros aspectos y mejorar así la experiencia del usuario en lugar de pelear por la reserva final sin cancelación.

Por último, dado que este proyecto se quiere realizar como una prueba de concepto para incluir a las industrias dentro del ecosistema sin incurrir en elevados gastos de estructura, personal o mantenimiento, se opta por alojarlo todo en un proveedor Cloud con opción a computación ServerLess. Esto permite conseguir un ecosistema escalable, seguro y de bajo coste.

Con todas estas premisas, se va a desarrollar el producto con un enfoque modular y flexible, permitiendo su adaptación a las necesidades específicas de diferentes usuarios y escenarios, asegurando así una solución robusta y útil para la industria hotelera.

Capítulo 3. OBJETIVOS

En este capítulo se establecen los objetivos del TFM, exponiendo tanto los objetivos generales como los específicos. Además, durante el capítulo se detallarán los beneficios esperados del trabajo.

3.1 Objetivos generales

Este proyecto se centra en tres objetivos generales:

OG.1 **Desarrollo de un modelo predictivo para paliar algún punto de dolor de la industria hotelera**: Crear un modelo de aprendizaje automático para que se puedan predecir las cancelaciones hoteleras, mejorando así la toma de decisiones operativas en la industria.

OG.2 **Despliegue de un sistema de gestión de modelos de aprendizaje automático (MLOps) en un entorno Cloud**: Implementar un sistema para monitorizar las diferentes etapas de un experimento de aprendizaje automático.

OG.3 **Creación de un producto colaborativo para los diferentes agentes de la industria**: Desarrollar un producto que utilice los modelos creados previamente y otros adicionales según demanda, incluyendo aspectos relacionados con la trazabilidad de las ejecuciones, los permisos de los usuarios, el consumo de los modelos y la integración con los sistemas propietarios de los hoteles.

La combinación de estos objetivos tiene como propósito crear un producto que terceros puedan usar para optimizar y mejorar sus operaciones comerciales, en un entorno seguro, trazado y en la nube.

3.2 Objetivos específicos

En esta sección se explican los objetivos específicos del proyecto, que van desde el análisis inicial de la industria hasta la implementación y el seguimiento de los modelos de aprendizaje automático. Estos objetivos están pensados para asegurar una comprensión profunda del problema y el desarrollo de soluciones tecnológicas.

OE.1 **Análisis de la industria y el contexto**: Realizar un análisis detallado de la industria hotelera y del problema de las cancelaciones de reservas para entender mejor el entorno y la viabilidad del proyecto.

OE.2 **Uso de la tecnología Cloud**: Implementar el uso de tecnología en la nube para aprovechar sus beneficios en el desarrollo y mantenimiento de toda la infraestructura subyacente.

OE.3 **Despliegue de un sistema de gestión de modelos de aprendizaje automático**: Configurar y desplegar un sistema que facilite la gestión de los modelos de aprendizaje automático, abarcando su desarrollo y evaluación.

OE.4 **Recopilación y preprocesamiento de datos**: Recopilar datos relacionados con la demanda de reservas hoteleras y realizar el preprocesamiento necesario para preparar

estos datos para el entrenamiento de modelos de aprendizaje automático.

OE.5 **Entrenamiento del modelo**: Desarrollar y entrenar modelos de aprendizaje automático para abordar el problema identificado, optimizando sus diferentes métricas.

OE.6 **Evaluación del modelo mediante el sistema de gestión de modelos de aprendizaje automático**: Utilizar el sistema de gestión para evaluar y comparar los modelos entrenados, eligiendo aquel que ofrezca los mejores resultados.

OE.7 **Creación de infraestructura para poder interactuar con diferentes modelos**: Desarrollar una infraestructura que permita a los usuarios interactuar con los diferentes servicios de la plataforma.

OE.8 **Sistemas de incentivos para fomentar la colaboración de los agentes**: Establecer un sistema de incentivos que promueva la colaboración de los diferentes agentes de la industria, recompensando su participación y contribución a mejorar el ecosistema.

OE.9 **Monitoreo y plan de reentrenamiento del modelo**: Implementar un sistema para monitorear el rendimiento del modelo y un plan para reentrenarlo periódicamente.

3.3 Beneficios del proyecto

Este proyecto ayudará a la industria hotelera a tomar decisiones más informadas mediante un modelo predictivo que anticipa las cancelaciones de reservas, lo que permitirá optimizar la gestión de recursos y maximizar los ingresos. El proyecto también promoverá la colaboración entre los diferentes agentes de la industria hotelera, permitiendo compartir datos y modelos de manera segura. Además, se asegurará la precisión y adaptabilidad de los modelos mediante el monitoreo y el reentrenamiento periódico.

Capítulo 4. DESARROLLO DEL PROYECTO

En este capítulo se describe cómo se desarrolla el proyecto. Primero, se presenta la planificación, los recursos necesarios y el presupuesto, seguido de un análisis de viabilidad. El cimiento de este desarrollo es la implementación de un sistema de gestión de modelos de aprendizaje automático en una infraestructura en la nube. Se explica el desarrollo del modelo para predecir cancelaciones de reservas hoteleras, desde la definición del problema hasta la evaluación de los modelos. Finalmente, se detalla el sistema de servicio de modelos de aprendizaje automático como pieza clave del TFM, y se presentan los resultados obtenidos del proyecto.

4.1 Planificación del proyecto

En esta sección se incluirá la planificación del proyecto, paso previo a cualquier tipo de desarrollo, análisis o evaluación. De esta forma se pueden acotar los tiempos de trabajo asignados a cada una de las tareas y actividades.

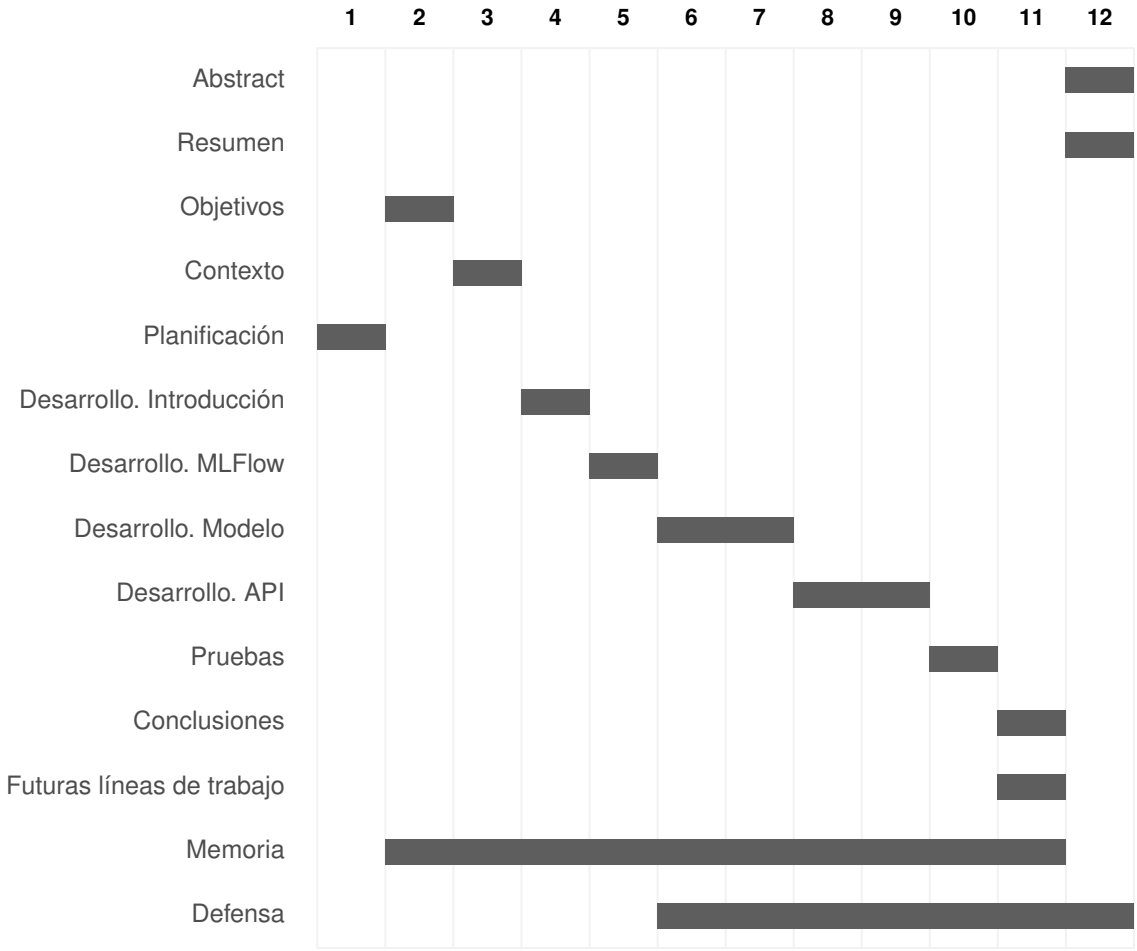

Tabla 4.1. Diagrama de Gantt de planificación del proyecto

Como se puede ver en la Tabla 4.1, este proyecto se desarrolla a lo largo de 12 semanas,

entre mayo y julio de 2024. El mes de abril se destina a concretar los detalles del proyecto con el tutor, agosto se considera no lectivo, y en septiembre se dedica tiempo a preparar la defensa.

4.2 Recursos requeridos

En esta sección se enumeran los recursos utilizados para la ejecución del proyecto.

- **Ordenador de sobremesa**:
  - Procesador: Ryzen 7 3700x
  - Memoria RAM: 16 GB
  - Tarjeta gráfica: Nvidia GTX 1660 Super
- **Ordenador portátil**:
  - Procesador: Intel i7-8550U
  - Memoria RAM: 8 GB
  - Tarjeta gráfica: Nvidia MX150
- **Monitores**:
  - Monitor de 27” con resolución 2K
  - Monitor de 24” con resolución 1080p
  - Monitor vertical de 22” con resolución 1080p
- **Otros periféricos**:
  - Teclado
  - Ratón
  - Auriculares
- **Servicios en la nube**:
  - Cuenta AWS Free Tier [9]
- **Software**:
  - IDE Visual Studio Code Insiders
  - WSL2 para Windows 11
  - Entorno virtual Python
  - Software MLFlow
- **Herramientas de redacción**:
  - Cuenta Overleaf para la redacción de la memoria del TFM
  - Acceso a PowerPoint para la defensa del TFM

4.3 Sistema de gestión de modelos de Machine Learning sobre infraestructura Cloud

El primer paso antes de entrenar un modelo es establecer un sistema de gestión de experimentos de aprendizaje automático. En este caso, se ha decidido utilizar MLflow [10] para llevar a cabo esta tarea. Además, para asegurar una mayor escalabilidad, accesibilidad y facilidad de uso, se ha implementado este sistema sobre una infraestructura en la nube.

Para desplegar este sistema de manera escalable y replicable, es fundamental utilizar infraestructura como código (IaC) [11]. Dado que AWS es la nube elegida, se ha decidido emplear AWS CDK [12], la herramienta nativa de AWS para esta tarea. Esta elección se prefiere sobre otras herramientas de terceros, como Serverless Framework o Terraform, por su integración

y compatibilidad nativas con AWS y el soporte del mismo proveedor.

El primer paso es crear una cuenta de AWS y configurar un usuario de IAM para obtener las claves necesarias para acceder a la nube utilizando AWS CLI. Una vez realizado este paso, es necesario realizar el bootstraping de la cuenta con AWS CDK [13]. Este proceso implica la configuración inicial de la cuenta de AWS para poder desplegar stacks con CDK. Esto incluye la creación de ciertos recursos necesarios, como roles de IAM y buckets de S3, que facilitan la gestión y despliegue de infraestructura mediante AWS CDK.

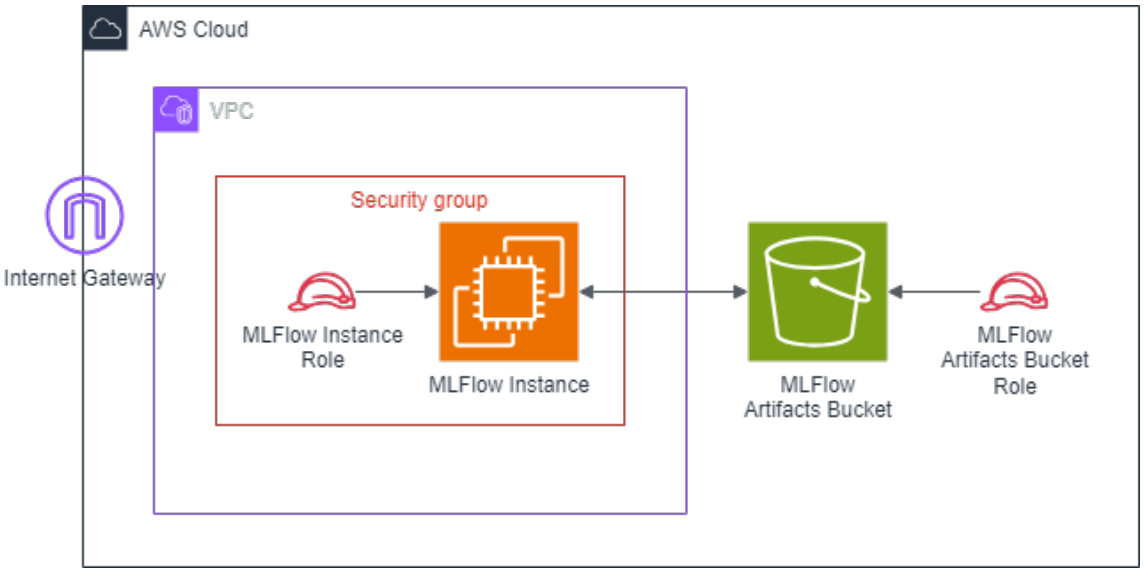

Figura 4.1. Diagrama arquitectura módulo MLFlow

Una vez se han completado los pasos para establecer el entorno de despliegue, es necesario crear un stack con los recursos definidos. Todos los recursos que se definan a lo largo de este proyecto estarán contenidos dentro de la capa gratuita de AWS. Esto se hace con el objetivo de minimizar los costos y demostrar que es posible crear productos mínimos viables para startups y validar ideas sin necesidad de realizar una inversión inicial elevada. Un diagrama de la arquitectura orientativo que se puede ver en la Figura 4.1.

Primero, se crea una VPC (Virtual Private Cloud) que proporciona un entorno de red aislado y seguro para alojar los recursos de la infraestructura. Luego, se define un bucket de S3 para almacenar los artefactos de MLflow. Este bucket incluye una política de recursos que permite el acceso completo a la cuenta de AWS para leer y escribir los artefactos.

Para controlar el tráfico y asegurar la instancia, se crea un grupo de seguridad. Este grupo permite el tráfico SSH, HTTP, HTTPS y el acceso a MLflow desde cualquier dirección IP, asegurando que la instancia de EC2 sea gestionable remotamente y que el servicio de MLflow esté accesible públicamente. Se define un rol de IAM que es asumido por las instancias de EC2, otorgándoles permisos para interactuar con el bucket de S3.

Finalmente, se crea una instancia EC2 en la VPC, configurada para instalar las dependencias necesarias, como Python y MLFlow, y para iniciar el servidor de MLFlow. Esta instancia cuenta con una dirección IP pública para garantizar su accesibilidad externa, como se muestra en

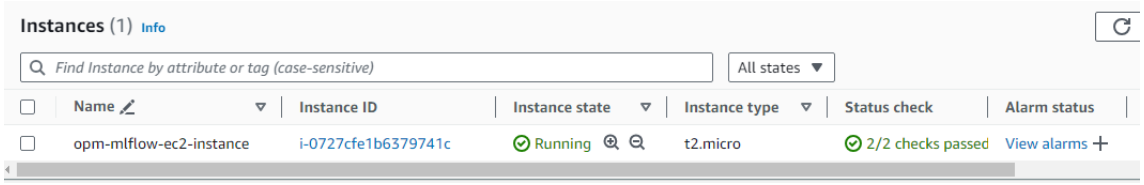

Figura 4.2. Instancia EC2 desplegada para MLFlow

la Figura 4.2.

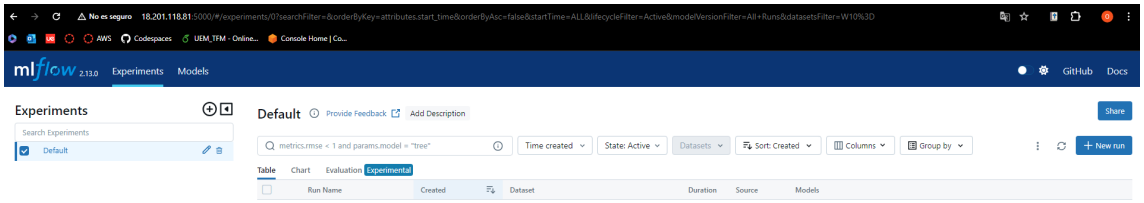

Figura 4.3. Interfaz gráfica de MLFlow (MLFlow UI)

Una vez que la instancia se ha iniciado y se han ejecutado los comandos definidos como user data, se puede acceder a ella a través de la IP pública expuesta o mediante la ruta Public IPv4 DNS a través del puerto 5000. Al acceder mediante un navegador, es posible ver la interfaz gráfica de MLFlow, como se muestra en la Figura 4.3.

Con el objetivo de probar el funcionamiento, se lleva a cabo el entrenamiento de un modelo de ejemplo. Durante este proceso, se registran las métricas, lo que permite elegir el mejor modelo, así como el propio modelo, para su uso posterior.

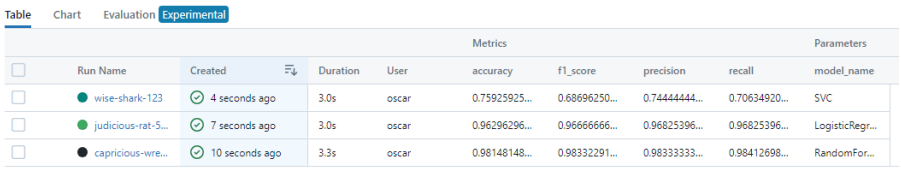

Figura 4.4. Ejecuciones de experimentos en MLflow

En este caso, se ha utilizado el dataset Wine de la biblioteca Sklearn [14], entrenando tres modelos diferentes: una regresión logística, un random forest y un SVC. Al realizar estos entrenamientos bajo el experimento por defecto, podemos observar los diversos registros en MLflow, como se aprecia en la Figura 4.4.

Además, es posible realizar comparaciones entre las métricas de los diferentes modelos utilizando la funcionalidad integrada de MLflow para mostrar gráficos, como se puede ver en la Figura 4.5. Esto facilita enormemente la evaluación de los modelos, permitiendo un seguimiento detallado de su rendimiento durante el entrenamiento.

Finalmente, se prueba la característica de servir modelos registrados en MLflow. Este proceso descarga los artefactos del bucket de S3 a través de la instancia de EC2 en el entorno local de Python. A continuación, se ejecuta la inferencia y se obtienen las predicciones, como se muestra en la Figura 4.6.

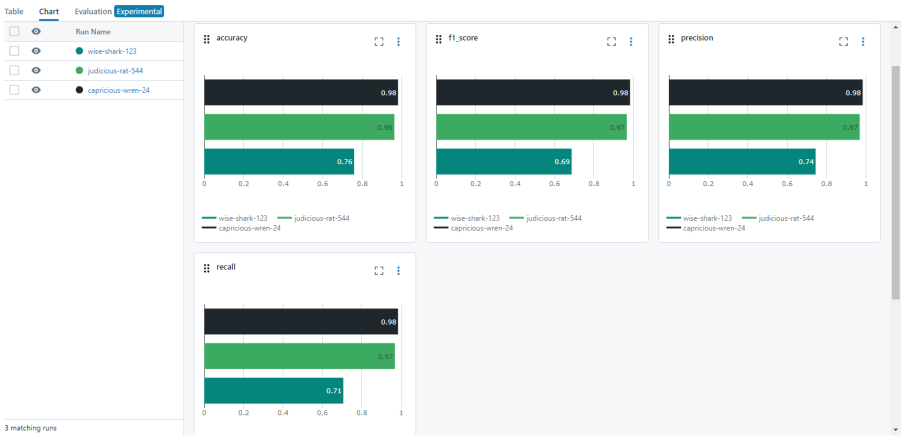

Figura 4.5. Gráficos de métricas de modelos en MLflow

```
logged_model = 'runs:/fa75fbcfcc4d4931a42c0375b1177fdf/RandomForest'
loaded_model = mlflow.pyfunc.load_model(logged_model)

wine = load_wine()
X = wine.data
feature_names = wine.feature_names

np.random.seed(42)
random_indices = np.random.choice(X.shape[0], 5, replace=False)
data = X[random_indices]

input_data = pd.DataFrame(data, columns=feature_names)

predictions = loaded_model.predict(input_data)

print("Predicciones:\n", predictions)
✓ 1.0s

Downloading artifacts: 100% 9/9 [00:00<00:00, 21.30it/s]

Predicciones:
 [0 0 2 0 1]
```

Figura 4.6. Prueba del uso de un modelo registrado en MLflow

4.4 Desarrollo modelo de predicción de cancelaciones de reservas hoteleras

Como en todo desarrollo de un modelo de aprendizaje automático, es necesario definir una serie de etapas para realizar un desarrollo estructurado. A continuación, se presentan las etapas más utilizadas en la industria [15] que permiten cumplir este objetivo:

1. **Definición del problema:** Esta etapa consiste en comprender y especificar claramente el problema que se quiere resolver con el modelo de aprendizaje automático. También se deben considerar las restricciones y el contexto en el que se aplicará el modelo.

2. **Obtención de los datos:** En esta fase se recopilan los datos necesarios para entrenar el modelo. Es crucial asegurarse de que los datos sean relevantes y actuales, en la medida de lo posible.

3. **Inspección y preparación inicial de los datos:** Aquí se realiza una primera exploración de los datos para entender su estructura y contenido. Esta etapa también incluye la identificación de datos faltantes o inconsistentes.
4. **Preparación de los datos para el modelo:** Esta fase implica la limpieza y transformación de los datos para hacerlos adecuados para el entrenamiento del modelo. Incluye tareas como la imputación de valores faltantes, la codificación de variables categóricas y la creación de nuevas características (*feature engineering*).
5. **Selección, entrenamiento y ajuste del modelo:** En esta etapa se seleccionan los algoritmos de aprendizaje automático más adecuados para el problema y se entrenan con los datos preparados. Se comparan diferentes modelos y se elige el mejor en función de las métricas.
6. **Puesta en producción y mantenimiento del modelo:** Una vez que se ha desarrollado y validado el modelo, se despliega en un entorno de producción donde puede generar predicciones. Esta etapa también incluye el monitoreo continuo del rendimiento del modelo y la actualización del mismo según sea necesario para mantener su eficacia con a lo largo del tiempo.

4.4.1. Definición del problema

En esta primera etapa es fundamental definir el objetivo de negocio a resolver. Este objetivo nunca debe ser el uso de un modelo o técnica específica, sino que debe ser una abstracción del negocio a un nivel más alto.

En este caso, el objetivo es predecir la cancelación de una reserva de hotel antes de que ocurra para poder tomar ciertas acciones de negocio. De este modo, el modelo se enmarcaría dentro de un sistema de aprendizaje supervisado y resolvería un problema de clasificación binaria, ya que solo hay dos categorías: cancelará o no cancelará.

Además, con el objetivo de convertir este modelo en un producto, es necesario que el sistema pueda ejecutarse en un entorno Cloud y Serverless para optimizar costos. También debe permitir el reentrenamiento para corregir posibles desviaciones del modelo sin que esto implique un modelo extremadamente complejo y difícil de mantener con un número elevado de características.

Finalmente, el modelo debe ser entrenable en un ordenador convencional, sin la necesidad de una gran capacidad de cómputo distribuida, como por ejemplo LORCA, el que se ofrece desde la propia escuela [16]. Además, debe poder realizar inferencia en un entorno Cloud.

4.4.2. Obtención de los datos

El modelo inicial se basará en los datos publicados en la revista 'Data in Brief', en el artículo 'Hotel booking demand datasets' por Nuno Antonio, Ana de Almeida y Luis Nunes [17].

Los datos utilizados en el estudio provienen de dos tipos de establecimientos hoteleros: hotel resort (H1) en la región de Algarve y un hotel urbano (H2) en Lisboa. Estos datos fueron extraídos de los sistemas de gestión de propiedades de los hoteles (PMS) mediante consultas SQL, y contienen observaciones de reservas entre el 1 de julio de 2015 y el 31 de agosto de 2017, abarcando reservas tanto completadas como canceladas. La estructura de los conjuntos de datos incluye 31 variables que describen un total de 40,060 observaciones para el hotel H1 y 79,330 para el hotel H2.

Además, para garantizar la calidad y confidencialidad de los datos, toda información que pudiera identificar a hoteles o clientes fue eliminada por los autores del artículo. Los datos fueron analizados usando el lenguaje de programación R, y están disponibles junto con el artículo para su uso en investigación y educación.

Estos conjuntos de datos son significativos debido a la escasez de datos empresariales reales disponibles para propósitos científicos y educativos, ofreciendo una base rica para investigaciones futuras en la gestión de demanda hotelera y en la predicción de cancelaciones de reservas. Esta situación no es ideal, ya que las cadenas hoteleras no suelen disponer de conjuntos de datos públicos para este tipo de uso, debido al valor estratégico de los mismos para su negocio. Por esta razón, es necesario obtenerlos de fuentes públicas que hayan conseguido estos datos a través de alguna colaboración o hayan sido donados de manera desinteresada.

Además, el proyecto no solo pretende utilizar los datos estáticos del conjunto de datos para realizar el modelo, sino que se propone un modelo de aprendizaje automático dinámico, que recibe datos continuamente de los diferentes usuarios del sistema a cambio de beneficios económicos por su uso. Para garatizar la independencia de los actores, se considera siempre que estos datos no serán accesibles por otros agentes y que únicamente se utilizarán para reentrenar el modelo de manera periódica.

4.4.3. Inspección y preparación inicial de los datos

En esta etapa, el principal objetivo es importar los datos al entorno de trabajo, analizar cada una de las columnas para comprenderlas, identificar posibles relaciones entre variables, detectar valores faltantes o duplicados y realizar una preparación inicial de los datos en bruto. Todo esto con el fin de estar preparados para las siguientes etapas del proceso de Aprendizaje Automático.

Como primera acción, es necesario importar los dos archivos que componen el estudio: 'H1.csv' y 'H2.csv'. En este paso, se comprueba que ambos dataframes tienen las mismas columnas y se procede a estudiar cada una de ellas, basándose en la información empírica de la importación y en la información contenida en el artículo de referencia.

1. **ADR** (Numérica): Tarifa diaria promedio.
2. **Adults** (Entera): Número de adultos.
3. **Agent** (Categórica): ID de la agencia de viajes que hizo la reserva.

4. **ArrivalDateDayOfMonth** (Entera): Día del mes de la fecha de llegada.
5. **ArrivalDateMonth** (Categórica): Mes de la fecha de llegada.
6. **ArrivalDateWeekNumber** (Entera): Número de semana de la fecha de llegada.
7. **ArrivalDateYear** (Entera): Año de la fecha de llegada.
8. **AssignedRoomType** (Categórica): Código del tipo de habitación asignada a la reserva.
9. **Babies** (Entera): Número de bebés.
10. **BookingChanges** (Entera): Número de modificaciones realizadas a la reserva desde el momento en que se ingresó en el PMS hasta el momento del check-in o cancelación.
11. **Children** (Entera): Número de niños.
12. **Company** (Categórica): ID de la entidad que hizo la reserva.
13. **Country** (Categórica): País de origen. Las categorías se representan en el formato ISO 3155–3:2013 según [18].
14. **CustomerType** (Categórica): Tipo de reserva.
15. **DaysInWaitingList** (Entera): Número de días que la reserva estuvo en la lista de espera antes de ser confirmada al cliente. Calculado restando la fecha en que la reserva fue confirmada al cliente de la fecha en que se ingresó en el PMS.
16. **DepositType** (Categórica): Indicación de si el cliente hizo un depósito para garantizar la reserva.
17. **DistributionChannel** (Categórica): Canal de distribución de la reserva.
18. **IsCanceled** (Categórica): Valor que indica si la reserva fue cancelada (1) o no (0).
19. **IsRepeatedGuest** (Categórica): Valor que indica si el nombre de la reserva pertenece a un huésped recurrente (1) o no (0).
20. **LeadTime** (Entera): Número de días transcurridos entre la fecha de ingreso de la reserva en el PMS y la fecha de llegada.
21. **MarketSegment** (Categórica): Designación del segmento de mercado.
22. **Meal** (Categórica): Tipo de comida reservada.
23. **PreviousBookingsNotCanceled** (Entera): Número de reservas anteriores no canceladas por el cliente antes de la reserva actual.
24. **PreviousCancellations** (Entera): Número de reservas anteriores que fueron canceladas por el cliente antes de la reserva actual.
25. **RequiredCardParkingSpaces** (Entera): Número de espacios de parking requeridos por el cliente.
26. **ReservationStatus** (Categórica): Último estado de la reserva.
27. **ReservationStatusDate** (Fecha): Fecha en que se estableció el último estado.

28. **ReservedRoomType** (Categórica): Código del tipo de habitación reservada.
29. **StaysInWeekendNights** (Entera): Número de noches de fin de semana que el huésped se quedó o reservó para quedarse en el hotel.
30. **StaysInWeekNights** (Entera): Número de noches entre semana que el huésped se quedó o reservó para quedarse en el hotel.
31. **TotalOfSpecialRequests** (Entera): Número de solicitudes especiales hechas por el cliente.

Se ha comprobado que ambos DataFrames tienen las mismas columnas. Para simplificar el trabajo, se han unido en un único DataFrame, añadiendo una columna adicional para identificar si el hotel es de tipo ciudad o resort. Tras la concatenación de los dos DataFrames, se ha restablecido el índice de cada registro.

Se ha observado que en el dataset inicial existen valores con espacios al inicio o al final del texto, corrigiéndose de manera sencilla con un Regex.

Se han analizado los valores faltantes en el dataset, encontrando que las columnas 'Country' y 'Children' tienen 488 y 4 valores faltantes respectivamente, lo que corresponde al 0.4 % y 0.003 %. Para los valores faltantes en 'Country', se ha asignado 'Portugal', dado que los datos provienen de una cadena hotelera portuguesa. Para los valores faltantes en 'Children', se ha asignado el valor 0, asumiendo que la reserva no incluye niños.

Se ha llevado a cabo una comparación entre la descripción de las variables y los datos empíricos proporcionados. Se identificaron discrepancias en las siguientes variables: 'Children', que se describió como un entero pero en los datos aparece como float64. Por otro lado, 'ReservationStatusDate', que se describió como una fecha pero en los datos es de tipo object. Se corrigen estas disparidades.

Se han estimado las duplicidades en el dataset, pero no se ha tomado ninguna medida correctora debido a que dentro del marco del problema estas duplicidades pueden existir y ocasionar un peso mayor de los registros iguales sobre el modelo final.

4.4.4. Preparación de los datos para el modelo

El primer paso en la preparación de los datos para el modelo consiste en eliminar y realizar ingeniería de variables [19] para descartar o modificar aquellas que no sean útiles para la predicción.

En primer lugar, las variables 'Assigned Room Type' y 'Reserved Room Type' clasifican cada tipo de habitación con valores alfabéticos. Dado que no disponemos de información sobre el tipo específico que representa cada valor (interior, balcón, exterior, etc.), estas variables no aportan información útil. Sin embargo, al combinar ambas y aplicar ingeniería de características, se puede crear una nueva variable, 'IsRoomTypeAvailable'. Esta nueva variable indicará si la habitación reservada es la misma que la asignada finalmente y puede utilizarse para evaluar la disponibilidad de las habitaciones.

La variable 'Country' presenta 177 valores únicos, lo que implicaría la creación de 177 nuevas variables al realizar la transformación para el entrenamiento. Para optimizar este proceso, se decide agrupar ciertos países. En el análisis realizado, se observó que los países Portugal (41.1 %), Reino Unido (10.2 %), Francia (8.7 %), España (7.2 %), Alemania (6.1 %), Italia (3.2 %) e Irlanda (2.8 %) tienen los mayores porcentajes. Con el objetivo de disminuir la dimensionalidad del dataset, se decide conservar únicamente los países que representan más del 5 % y agrupar el resto en una categoría única. Esta nueva variable Other representaría el 26 % de los registros.

Se verifica que la suma de los valores en las variables 'Adults', 'Babies' y 'Children' sea mayor que cero para asegurar que haya al menos una persona en cada reserva. En este proceso, se identifican 180 registros, que representan menos del 0.1 % del total. Estos registros no son válidos y, por lo tanto, se eliminan del conjunto de datos.

Se verifica que los valores en las variables 'StaysInWeekendNights' y 'StaysInWeekDays' sean mayores que cero para asegurar que haya al menos una noche de estancia en cada reserva. En este proceso, se identifican 645 registros con ambos valores en cero. Estos registros no son válidos y se eliminan del conjunto de datos.

Se convierte la variable 'Agent' en 'IsAgentBooking', ya que 'Agent' muestra el identificador del agente como un valor numérico, y un valor NULL para aquellos registros que no han sido reservados por un agente. Esta nueva variable dicotómica, 'IsAgentBooking', indica si la reserva fue realizada a través de un agente, lo que permite utilizar esta información para realizar predicciones.

Al igual que ocurre con la variable 'Agent', se convierte la variable 'Company' en una nueva variable llamada 'IsCompanyBooked'. Al crear 'IsCompanyBooked', se puede identificar fácilmente si una reserva fue realizada a través de una empresa, lo que facilita su uso en predicciones.

También se transforma la variable 'ArrivalDateMonth', que es categórica y contiene los meses del año como texto, en valores numéricos, realizándose un mapeo de cada mes a su correspondiente valor numérico.

Dado que ya existe la variable 'IsCanceled', se eliminan las variables 'ReservationStatus' y 'ReservationStatusDate', ya que no aportan información adicional.

Como último paso, y para que los diferentes algoritmos de aprendizaje automático que se prueben puedan ejecutarse, se realiza una codificación 'One-Hot Encoding' [20] para convertir las variables categóricas en variables numéricas. Se elige la columna a excluir, la menos representativa, con valores como 'Other' o 'Undefined'. Este paso aumenta las dimensiones del dataset de 29 variables a 48.

Una vez realizadas todas estas transformaciones, se procede a exportar un CSV, el cual se utilizará en el siguiente paso de selección, entrenamiento y ajuste de los modelos.

4.4.5. Selección, entrenamiento y ajuste del modelo

Una vez creado el dataset de entrenamiento, se pasa a la siguiente etapa de selección, entrenamiento y ajuste del modelo. En esta etapa, el objetivo es crear un modelo de aprendizaje automático que sea capaz de predecir las cancelaciones hoteleras.

MLFlow

Para esto, se hace uso del sistema de MLFlow desplegado previamente en AWS. Para facilitar el registro de los modelos, se crea una función para registrar cada uno de los modelos. En esta función, primero se configura el experimento y se inicia una nueva ejecución de un modelo. Con esto, se registran los parámetros del modelo y el dataset usado, así como el nombre del modelo para poder identificarlo de manera correcta. A continuación, se entrena el modelo, se realizan las predicciones y finalmente se calculan y registran tanto las métricas del modelo como el modelo final. Esta función se pude ver en el Listing 4.1.

Listing 4.1: Función registro modelos MLFlow

```
def log_model_to_mlflow(model, df, model_name, X_train,
y_train, X_test, y_test, params, experiment_name=EXPERIMENT_NAME):
    mlflow.set_experiment(experiment_name)
    with mlflow.start_run():
        mlflow.log_params(params)
        dataset = mlflow.data.from_pandas(
            df, name="hotel_reservation", targets="IsCanceled"
        )
        mlflow.log_input(dataset)
        mlflow.log_param("model_name", model_name)
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        metrics = {
            "accuracy": accuracy_score(y_test, y_pred),
            "precision": precision_score(y_test, y_pred),
            "recall": recall_score(y_test, y_pred),
            "f1_score": f1_score(y_test, y_pred),
            "roc_auc": roc_auc_score(y_test, y_pred),
        }
        mlflow.log_metrics(metrics)
        signature = infer_signature(X_test, y_pred)
        mlflow.sklearn.log_model(model, "model", signature=signature)
        mlflow.end_run()
```

Una vez creada está función se establece el nombre del experimento por defecto, en este caso 'cancelation-prediction' y la url del servidor de MLFlow. Se importa el dataset y se crean los conjuntos de train y test.

Descripción de los modelos

Con esto se proponen los siguientes modelos de aprendizaje automático para poder hacer una primera criba y optimizar los algoritmos resultantes: Regresión Logística [21] (Logistic Regression), Clasificador de Vecinos Más Cercanos [22] (KNeighbors Classifier), Clasificador de Árbol de Decisión [23] (Decision Tree Classifier), Bosque Aleatorio [24] (Random Forest Classifier), Clasificador AdaBoost [25] (AdaBoost Classifier), Clasificador de Gradient Boosting [26] (Gradient Boosting Classifier), Clasificador XGBoost [27] (XGBClassifier), Clasificador CatBoost [28] (CatBoostClassifier), Clasificador LightGBM [28] (LGBMClassifier), Clasificador de Árboles Extra [29] (ExtraTrees Classifier) y Clasificador de Votación [30] (Voting Classifier). Estos modelos contienen muchas técnicas y enfoques diferentes para problemas muy conocidos, todos ellos optimizados por los creadores de las librerías: sklearn [31], xgboost[27], catboost [32] y lightgbm [28].

Es importante conocer de manera conceptual cada uno de los algoritmos antes de ejecutarlos, haciendo énfasis en aquellos seleccionados para estudio más adelante, por lo que se procederá a explicar cada uno, así como el grupo al que pertenecen.

El primero, la regresión logística, es uno de los más sencillos y se utiliza para obtener la probabilidad de que una observación pertenezca a una de las dos clases posibles. Por otro lado, se utiliza la técnica de clasificación de vecinos más cercanos, que no hace suposiciones sobre la distribución de los datos y clasifica las observaciones en base a sus k vecinos más cercanos, aunque tiene un alto coste computacional.

El clasificador por árbol de decisión es un algoritmo tanto de perdición como de clasificación ampliamente utilizado en el campo del aprendizaje automático. Su estructura se asemeja a un árbol, donde cada nodo interno representa una decisión sobre una característica, cada rama el resultado de una decisión, y cada hoja terminal representa una clase o una predicción. Este algoritmo se optimiza mediante la minimización de una función de coste, como el índice de Gini o la Entropía, que mide la pureza de los nodos. La idea es dividir los datos de tal manera que cada rama se acerque lo más posible a una clase pura, reduciendo la incertidumbre en las decisiones sucesivas.

Sin embargo, uno de los desafíos principales de los árboles de decisión es el problema del sobreajuste, donde el modelo se ajusta demasiado a los datos de entrenamiento y pierde capacidad de generalización a datos nuevos. Para mitigar este problema, se utiliza una técnica llamada bagging [33], que consiste en crear múltiples árboles de decisión a partir de diferentes subconjuntos de los datos originales, un bosque aleatorio [34]. Cada árbol del bosque se entrena de forma independiente y sus predicciones se promedian para obtener un resultado final. Esto no solo reduce el sobreajuste, sino que también mejora la precisión y la estabilidad del modelo al combinar múltiples perspectivas.

Además del bosque aleatorio, se considera el uso del clasificador de árboles extra, que es una variante que introduce un nivel adicional de aleatoriedad en la construcción de los árboles. Mientras que en el bosque aleatorio se seleccionan aleatoriamente los subconjuntos de datos y los atributos a considerar en cada división. En el clasificador de árboles extra, las divisiones mismas se seleccionan de forma completamente aleatoria, sin optimizar en base a ninguna

función de coste. Esta mayor aleatoriedad lleva a una mayor diversidad entre los árboles, resultando en un mejor rendimiento en ciertos problemas al evitar el sobreajuste de manera más efectiva que los bosques aleatorios tradicionales.

Por otro lado, se han probado los algoritmos de boosting [35], en concreto: AdaBoost, Gradiente Boosting, XGBoost y CatBoost. Estos algoritmos combinan varios clasificadores débiles para crear uno fuerte. En estos se van entrenando de manera secuencial, ponderando con un peso mayor aquellas observaciones mal clasificadas en cada iteración. Cada uno tiene características determinadas y busca solventar los errores de los otros. Por ejemplo, Gradient Boosting, en lugar de ponderar las observaciones mal clasificadas, optimiza la función de pérdida. XGBoost introduce regularización y paralelización para mejorar el rendimiento en grandes conjuntos de datos. CatBoost [36] maneja de manera correcta las variables categóricas sin transformaciones extensas. Finalmente, LightGBM busca la velocidad y el uso eficiente de la memoria, utilizando crecimiento basado en hojas y una reducción de datos por histogramas. Se probarán todos ellos para poder comparar.

Run Name	Create	Dataset	Duration	accuracy	f1_score	precision	recall	roc_auc
LogisticRegression	✓	hotel_r...	6.5s	0.76485472...	0.64343266...	0.75011182...	0.56331877...	0.72496881...
KNeighborsClassifier	✓	hotel_r...	9.2s	0.77366845...	0.67986877...	0.72747000...	0.63811443...	0.74684100...
AdaBoostClassifier	✓	hotel_r...	8.5s	0.82351452...	0.74560816...	0.81555851...	0.68670921...	0.79643944...
GradientBoostingClassifier	✓	hotel_r...	17.4s	0.84139501...	0.77699377...	0.82581295...	0.73362445...	0.82006618...
DecisionTreeClassifier	✓	hotel_r...	5.6s	0.85075696...	0.80221315...	0.80082570...	0.80360541...	0.84142522...
LGBMClassifier	✓	hotel_r...	5.4s	0.86159490...	0.80836155...	0.84466137...	0.77505318...	0.84446746...
VotingClassifier	✓	hotel_r...	31.4s	0.86463121...	0.80359765...	0.88587616...	0.73530399...	0.83903611...
CatBoostClassifier	✓	hotel_r...	15.7s	0.86893265...	0.81974248...	0.85031885...	0.79128876...	0.85356618...
XGBClassifier	✓	hotel_r...	5.3s	0.86893265...	0.82063711...	0.84673097...	0.79610345...	0.85451905...
ExtraTreesClassifier	✓	hotel_r...	22.3s	0.88284907...	0.83846958...	0.87214225...	0.80730041...	0.86789726...
RandomForestClassifier	✓	hotel_r...	19.1s	0.88622274...	0.84266386...	0.87927467...	0.80897995...	0.87093565...

Figura 4.7. Ejecuciones modelos registrados en MLFlow

Una vez hecho el entrenamiento se tienen todos los modelos registrados en MLFlow y con ello se van a seleccionar los 3 mejores para poder hacer una optimización más en profundidad. Las ejecuciones se pueden ver en la Figura 4.8.

Análisis de las métricas

Aun así para poder comparar todos los modelos mejor, se obtienen las gráficas para las cinco métricas registradas. Las cuales se pueden observar en las Figuras 4.8, 4.9, 4.10, 4.11 y 4.12.

Analizando estas métricas, se puede observar que todas tienen valores elevados, con los mejores modelos situándose entre 0.8 y 0.9. Se presta especial importancia al recall, ya que es la métrica que mide la proporción de verdaderos positivos frente a todos los casos reales positivos. En el contexto de las reservas hoteleras, es crucial minimizar el coste asociado a clasificar incorrectamente a usuarios que realmente cancelarán como si no fueran a cancelar, ya que esto impide prever y ajustar las ofertas adecuadamente para esos usuarios.

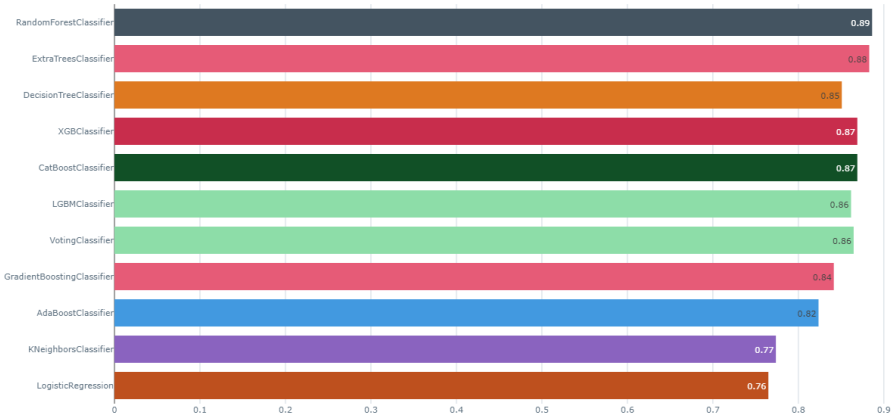

Figura 4.8. Comparación inicial accuracy

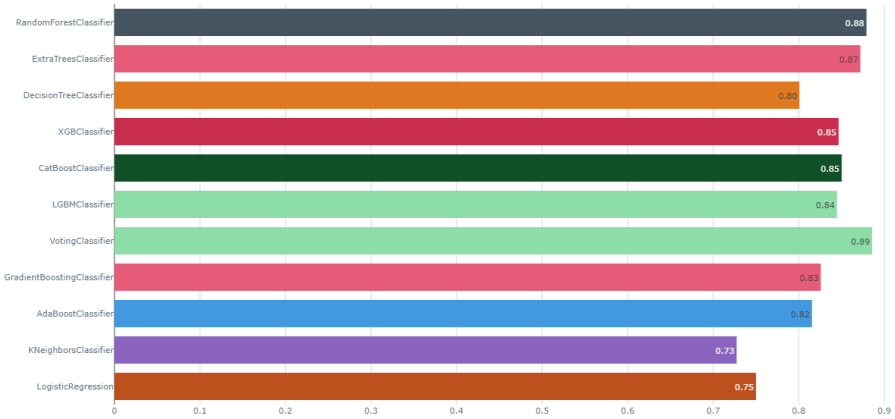

Figura 4.9. Comparación inicial precision

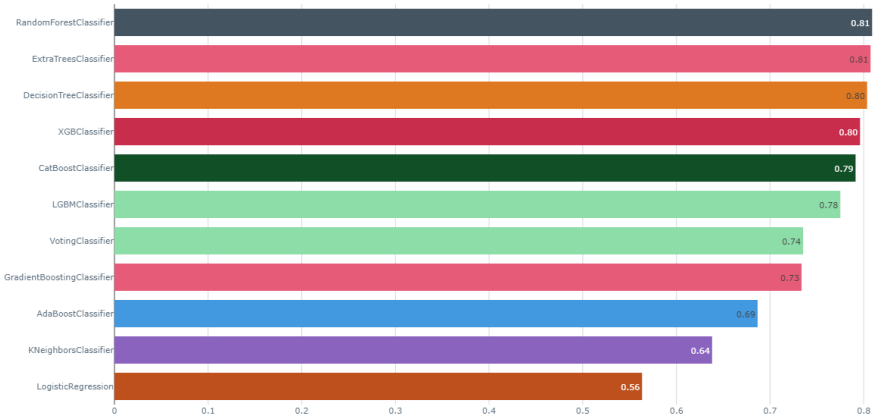

Figura 4.10. Comparación inicial recall

Además del recall, se ha considerado el F1 Score y el ROC AUC para obtener una evaluación más completa del rendimiento del modelo. El F1 Score proporciona un balance entre precisión y recall, y el ROC AUC mide la capacidad del modelo para distinguir entre clases.

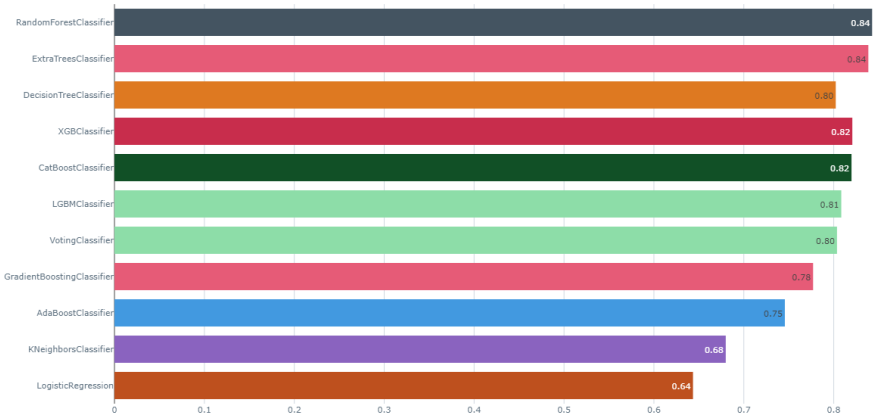

Figura 4.11. Comparación inicial f1 score

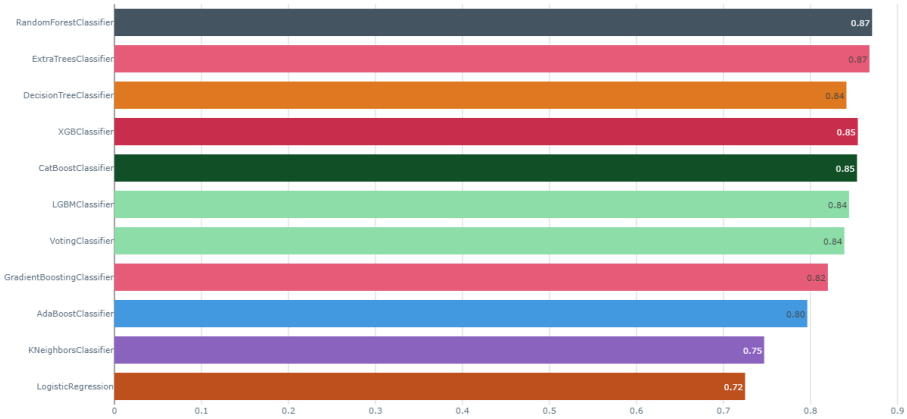

Figura 4.12. Comparación inicial roc auc

Por ello, los modelos RandomForestClassifier, ExtraTreesClassifier y DecisionTreeClassifier son los elegidos para llevar a cabo una evaluación más profunda, todos ellos perteneciendo a la misma familia de los clasificadores basados en árboles. Esto permite que también los modelos sean interpretables de manera sencilla, a comparación con los otros modelos de boosting, siendo los siguientes mejor clasificados.

Ajuste de hiperparámetros

Para poder realizar una optimización mayor sobre estos métodos, se ha procedido a realizar un BayesianSearchCV [37]. Este método optimiza los hiperparámetros mediante técnicas estadísticas bayesianas para encontrar el mejor conjunto de parámetros de manera más eficiente que GridSearchCV [38], así como encontrar valores dentro de un espectro continuo en lugar de tener que estar limitado dentro de la cuadrícula que se haya establecido.

Este método no explora todas las combinaciones posibles pero, dentro de un tiempo y unas iteraciones predefinidas, permite optimizar con un resultado similar a la búsqueda de hiperpa-

rámetros mediante cuadrícula. Además, este es más eficiente, más rápido y más inteligente, realizando menos evaluaciones.

Name	Depth	Split	Leaf	Boot	Est	Size
RF-BayesCV	30	2	1	False	500	1190 MB
RF	None	2	1	False	100	200 MB
ET-BayesCV	35	2	1	None	500	1777 MB
ET	None	2	1	None	100	422 MB
DT-BayesCV	26	1	2	None	None	7 MB
DT	None	1	2	None	None	1.7 MB

Tabla 4.2. Parámetros y tamaño de los diferentes clasificadores

De esta forma se han obtenido los siguiente modelos, con diferentes parámetros, pudiéndose ver en la Tabla 4.2, en este se puede ver como se ha llegado al máximo de número de estimadores y se ha elegido la configuración que más separa todos los valores. Además, se puede ver la diferencia en tamaño de los diferentes modelos, punto esencial a la hora de llevar un modelo a producción y tener que contabilizar el coste. En una situación ideal, no haría falta preocuparse por este tamaño pero como el objetivo de este modelo es poder servirse mediante un servicio de API, hay que tenerlo en cuenta a la hora de elegir el modelo a servir.

Name	Dur	Acc	F1	Prec	Rec	ROC
RF-BayesCV	33.8m	0.8861	0.0843	0.8743	0.8147	0.8720
RF	19.1s	0.8862	0.8427	0.8793	0.8090	0.8709
ET-BayesCV	24.8m	0.0883	0.8386	0.8710	0.8085	0.8681
ET	22.3s	0.8828	0.8385	0.8721	0.8073	0.8679
DT-BayesCV	3.6m	0.8530	0.8039	0.8071	0.8007	0.0843
DT	5.6s	0.8508	0.8022	0.8008	0.8036	0.8414

Tabla 4.3. Métricas de los diferentes clasificadores

Por otro lado, en la tabla de métricas se puede apreciar cómo todos los modelos mejoran ligeramente la métrica del recall, así como el resto de métricas. Además, es importante mencionar que en la de árbol de decisión no se ha conseguido mejorar el recall, por la poca variabilidad del árbol y de la situación inicial, pero el cambio es tan menor que no es relevante.

También, como paso esencial, se realiza un análisis de la importancia de cada una de las variables, como se puede ver en la Figura 4.13. En esta se puede ver que el tipo de depósito no reembolsable es una de las características más importantes y las que se pueden intuir de forma sencilla, como realiza el modelo de árbol de decisión. Otros factores importantes son: el tiempo de anticipación de la reserva, el país de procedencia de la reserva, el coste medio por noche y el número de solicitudes especiales del cliente. También se puede ver que el día y el mes de inicio de la reserva tienen una importancia moderada, lo que sugiere que existen momentos en el tiempo en los que es más probable la cancelación.

Estos datos y los futuros que se puedan ir generando son también útiles para los hoteles, con el fin de incentivar a los usuarios a no cancelar alterando algunas de las características

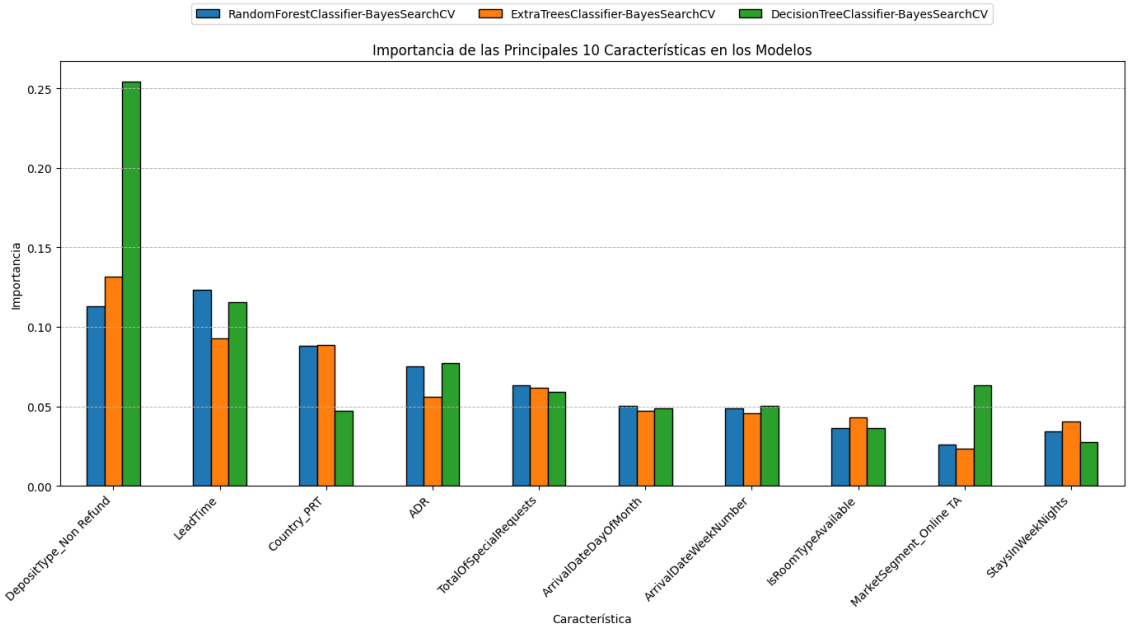

Figura 4.13. Importancia de las características de los modelos

más significativas. Por lo que de este análisis no se obtiene únicamente un modelo, sino un conocimiento extenso del problema y de cómo paliarlo.

Resulta complicado elegir uno, ya que es necesario optimizar tanto las métricas del modelo como las características de entrenamiento y memoria al servirse por API. Por lo tanto, dado que el servicio permite utilizar diversos modelos en la misma API, se toma la decisión de seleccionar, maximizando las métricas de predicción, el bosque aleatorio grande generado mediante búsqueda bayesiana. Por otro lado, para optimizar el uso de memoria y el tamaño del modelo, se optará por el árbol de decisión sencillo. De esta forma se han elegido dos modelos, que podrán ser usados de forma sencilla.

4.4.6. Puesta en producción y mantenimiento del modelo

Una vez completados todos los pasos del desarrollo de un modelo de aprendizaje automático, es crucial ponerlo en producción y establecer métodos de acceso estandarizados. Este proceso se llevará a cabo durante el desarrollo del Sistema de servicio de modelos de Aprendizaje Automático descrito en la Sección 4.5. La implementación de estos métodos garantiza que el modelo pueda ser utilizado de manera eficiente y escalable en un entorno de producción, facilitando su integración con otros sistemas y aplicaciones, así como proveer de funciones adicionales como la gestión de usuarios, el consumo de créditos y la colaboración activa entre los diferentes agentes de la industria hotelera.

4.5 Sistema de servicio de modelos de Machine Learning

Una vez llevado a cabo el proceso de desarrollo del modelo de predicción de cancelaciones de reservas hoteleras en la sección anterior, se procede a realizar el servicio en la nube que permitirá el servicio vía API de todos los modelos creados.

4.5.1. Análisis y diseño de la solución

El primer paso es analizar y conocer los requisitos y el diseño de la solución, para poder proceder con la implementación del mismo.

Este sistema tendrá varios módulos como microservicios, con el objetivo de desacoplar en todos ellos las unidades de cómputo y únicamente compartir la infraestructura subyacente.

Estos módulos son los siguientes:

1. **Módulo de gestión de usuarios:** Incluye funcionalidades como la creación de usuarios, confirmación de cuentas, inicio de sesión y restablecimiento de contraseñas.
2. **Módulo de gestión de créditos y pagos:** Permite la consulta de créditos y la agregación de créditos por parte de un administrador o plataforma de validación de pagos.
3. **Módulo de gestión de modelos:** Facilita la creación, actualización y obtención de modelos de aprendizaje automático.
4. **Módulo de gestión de ejecuciones:** Se encarga de la gestión de las diferentes ejecuciones de cada modelo realizadas por los usuarios, permitiendo realizar predicciones, visualizar las ejecuciones y enviar el valor verdadero de las predicciones una vez se obtenga.
5. **Módulo de estadísticas:** Proporciona diversas estadísticas del uso del sistema.

Como se puede ver en la Figura 4.14, el sistema estará completamente alojado dentro de AWS. En esta figura se puede observar cómo existen diferentes recursos para satisfacer cada una de las necesidades del servicio: cómputo, almacenamiento, base de datos, entre otros, representados por cada uno de los servicios propios de AWS. Todas las peticiones de los usuarios llegarán a través de internet a una distribución de CloudFront, que, en este caso, apuntará a una URL propia de AWS, eliminando la necesidad de comprar un dominio y establecer los valores DNS correspondientes. Una vez que se acceda, el tráfico se dirigirá a AWS API Gateway, con el objetivo de distribuir el tráfico entre las diferentes unidades de cómputo y servicios. Además de controlar el redireccionamiento, en este paso se agrega la autenticación para invocar las API utilizando un token de acceso.

Asimismo, se pueden ver las diferentes Lambdas para cada uno de los servicios: créditos, modelos, ejecuciones, estadísticas y el módulo de Cognito para los usuarios. Todas estas unidades de cómputo realizan toda la lógica de negocio asociada al servicio y se comunicarán con los diferentes módulos de almacenamiento (Artifacts de MLFlow) y de base de datos (tabla de créditos, ejecuciones y modelos), todo ello con permisos personalizados y diversos métodos de acceso.

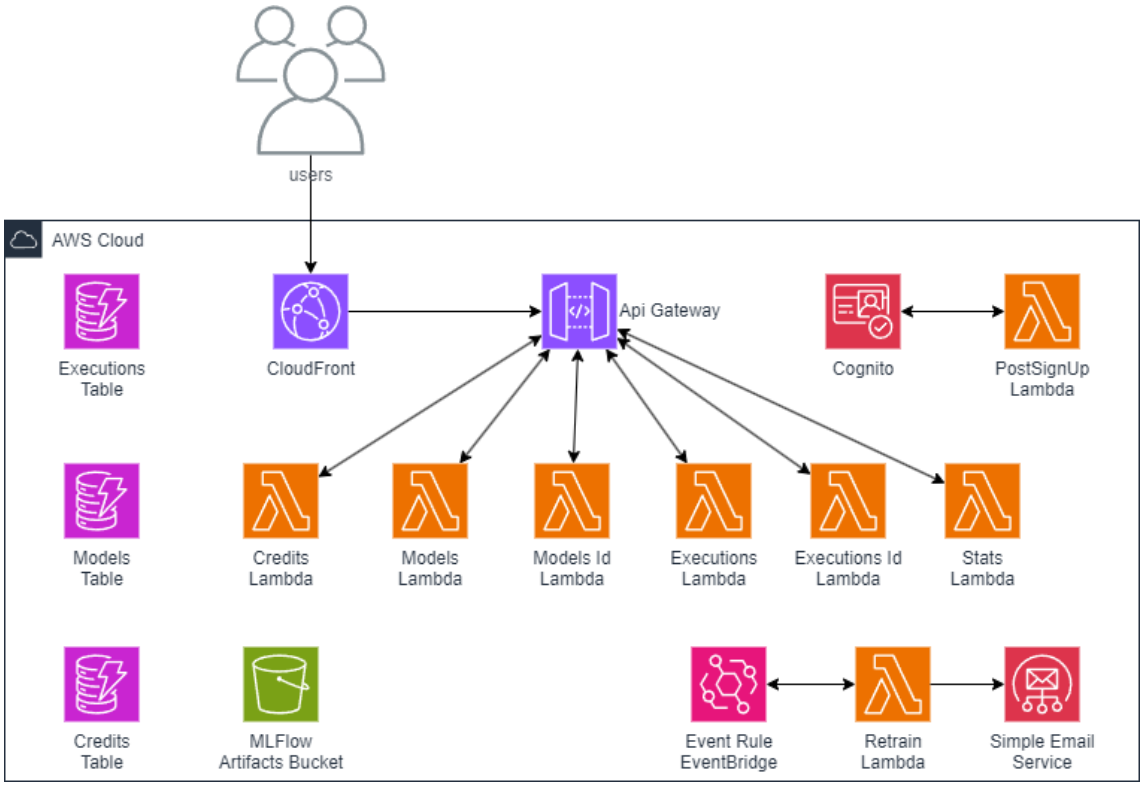

Figura 4.14. Diagrama de arquitectura general

Finalmente, se pueden apreciar una serie de Lambdas adicionales para controlar la inicialización de las tablas al momento de crear un nuevo usuario, así como una regla de tiempo que ejecutará una Lambda para recalcular las métricas de los modelos y enviar un correo electrónico avisando al operario para su reentrenamiento.

Además, es necesario analizar el modelo de datos para poder estudiar las necesidades y requerimientos de este, como se puede ver en la Figura 4.15. En este caso, las tablas estarán desplegadas sobre AWS DynamoDB, un sistema gestor de bases de datos no relacional de tipo documental, excepto la tabla de usuarios que está directamente sobre el servicio gestionado de AWS Cognito.

La tabla de usuarios contendrá el nombre de usuario, que será el email para evitar duplicidades de usuario y reducir también el almacenamiento de la información. También incluirá el sub como identificador único propio de Cognito, el perfil asignado al usuario, indicando si es administrador o usuario común, así como los valores de fecha de creación y fecha de actualización.

Relacionada con esta tabla de usuarios y para tener mayor control sobre la misma se crea en AWS DynamoDB la tabla de créditos. En esta tabla, el nombre de usuario será la clave única y contendrá el número de créditos actuales, así como el historial de pagos y el historial de ejecuciones. De esta forma, se podrá trazar cada uno de los usos de créditos del sistema y la agregación de nuevos créditos.

La tabla de modelos contendrá toda la información del modelo, con un identificador UUID úni-

Figura 4.15. Diagrama de datos general

co, el nombre del modelo, una pequeña descripción y el algoritmo que representa. También contendrá las métricas del modelo para poder compararlo posteriormente con las métricas empíricas, así como la información proporcionada por MLFlow para llevar a cabo la trazabilidad del entrenamiento del modelo (ruta del archivo del modelo, identificador del experimento e identificador del entrenamiento). Además, se agregan las variables para indicar cuántos

créditos consume la inferencia del modelo y cuántos créditos devuelve el hecho de proporcionar un valor verdadero. Finalmente, se incluyen otros campos como el nombre de usuario de quien creó el modelo, los campos para indicar la fecha de creación del modelo y el esquema de datos que hay que introducir al modelo.

La última estructura de datos es la tabla de ejecuciones, la cual contiene una clave única para cada inferencia y la clave del modelo que realiza la inferencia. Contiene además el nombre de usuario que realiza la inferencia, la fecha de realización de la inferencia, la información que el usuario introduce al modelo y el valor que este proporciona. Además, para poder reentrenar y estudiar la deriva del modelo, se agrega una columna más para guardar el valor real del suceso.

4.5.2. Módulo de gestión de usuarios

El módulo de gestión de usuarios se encarga de todo lo relacionado con el manejo de los usuarios y la correcta autenticación de los mismos para poder interactuar con el sistema. En este módulo se implementan acciones como iniciar sesión, crear cuenta, confirmar cuenta por correo electrónico y restablecer contraseña. Un diagrama de arquitectura más específico se puede ver en la Figura 4.16.

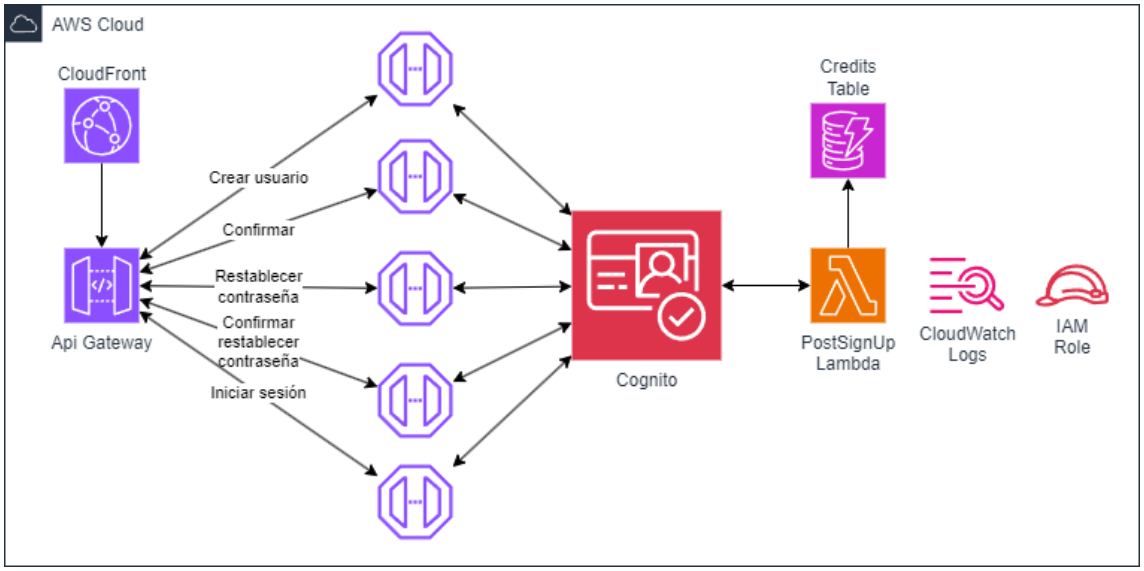

Figura 4.16. Diagrama de arquitectura del módulo de gestión de usuarios

El primer paso es crear el soporte y el servicio sobre los que se llevará a cabo el desarrollo de este módulo, utilizando AWS Cognito. Este servicio gestionado facilita y abstrae al desarrollador de mucha lógica de negocio relacionada con los usuarios, así como cumple con una gran cantidad de estándares de seguridad y diseño de serie. Primeramente, se crea el pool de usuarios, especificando ciertas opciones como la forma de iniciar sesión, la verificación de los atributos o permitir que un usuario pueda registrarse de manera libre. Por otro lado, para interactuar con este pool se crea el cliente, el cual va a soportar diversos métodos de autenticación, así como el dominio para tener un punto de acceso a este cliente. Con todos

estos recursos se procede a construir la API que integrará todo de forma sencilla.

Para construir la API se utiliza AWS API Gateway, y para realizar la integración se utilizan las opciones proporcionadas por AWS. De esta forma, se generan métodos para las diferentes acciones. La primera funcionalidad a codificar es la relativa a la creación de los usuarios, indicando los atributos requeridos (email y contraseña) y devolviendo un 200 OK en caso de que se haya creado el usuario correctamente o un 400 con un mensaje de error en caso de que no haya sido posible. La especificación de esta API se puede ver en el Anexo 8.1. Una vez creado el usuario, es necesario confirmar la propiedad del correo electrónico, para lo cual se envía un código numérico de 6 dígitos, que el usuario deberá introducir en los momentos posteriores a la creación del usuario. Este método devolverá un 200 OK en caso de que se haya confirmado y un 400 con el mensaje de error correspondiente en caso contrario. Igualmente, la especificación de esta API se puede ver en el Anexo 8.2.

En caso de que el usuario haya olvidado o necesite cambiar la contraseña, se establece el método de restablecer contraseña. Para esto, el usuario pedirá primero el cambio, recibiendo un código similar al mencionado anteriormente en el correo electrónico e introduciéndolo junto con la nueva contraseña en la llamada de confirmación de restablecimiento de contraseña, quedando la contraseña restablecida. Estas dos llamadas de API se pueden ver en los Anexos 8.3 y 8.4.

Finalmente, se establece el método de inicio de sesión en el que el usuario introducirá el nombre de usuario y la contraseña. Si todo ha ido correctamente, recibirá los tokens de acceso y un estado 200 OK, y en caso contrario un 400 con el correspondiente mensaje de error, como se puede ver en la especificación de API del Anexo 8.5. Además, se ha habilitado la opción de protección contra ataques por repetición para evitar que un ciberdelincuente pueda probar combinaciones hasta encontrar la opción adecuada. En esta respuesta, el AccessToken permite autenticar solicitudes a la API en nombre del usuario, el campo ExpiresIn muestra la duración del token en segundos, el campo IdToken contiene información del usuario en formato JWT, permitiendo así identificar la sesión del usuario y será usado para la autenticación del usuario en los servicios, y finalmente, el RefreshToken permite obtener nuevos tokens sin re-autenticarse y dura más que el AccessToken.

Además de todas estas APIs, se crea una Lambda como disparador cada vez que se crea un usuario nuevo para poder inicializar las diferentes estructuras de datos y establecer el contador de créditos a cero.

4.5.3. Módulo de gestión de créditos y pagos

Este segundo módulo de gestión de créditos y pagos es el encargado de realizar todo lo relacionado con la facturación y el plan de uso para los diferentes usuarios. En este se implementan los métodos de agregar créditos, ver los créditos disponibles, así como el histórico de uso de créditos. Finalmente, el método para poder sustraer créditos cuando se realicen las diferentes operaciones. Un diagrama específico de este módulo se puede ver en la Figura 4.17.

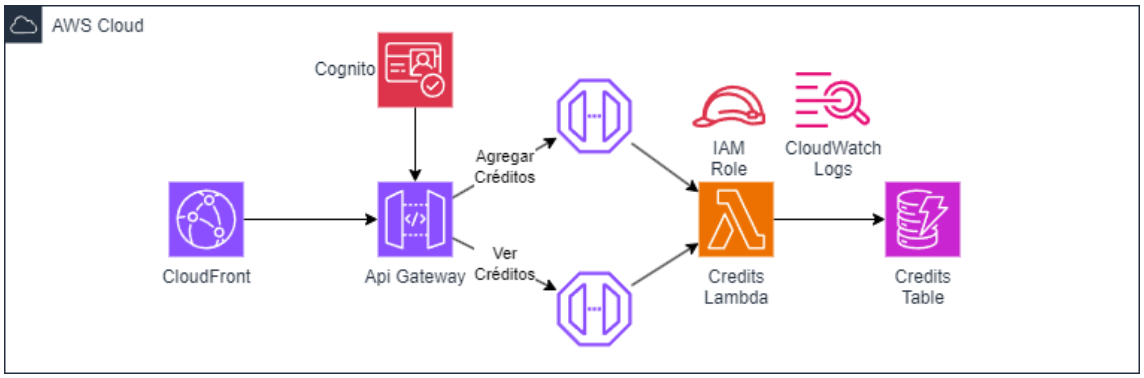

Figura 4.17. Diagrama de arquitectura del módulo de gestión de créditos y pagos

Este módulo contiene la tabla de créditos, la cual se crea sobre AWS DynamoDB y una Lambda para poder manejar la lógica de negocio. Así también se incluye, sobre la estructura de AWS API Gateway anterior, los distintos métodos que se podrán ejecutar sobre este módulo, todos ellos autenticados utilizando el token de acceso obtenido en el login.

La primera acción es la relativa a la consulta de créditos. Esta devuelve un 200 OK si se ha podido obtener tanto el número de créditos como las diferentes estructuras de datos que contiene, o un 404 en caso de que no haya podido encontrarse el usuario o haya habido algún problema con los permisos o el servidor. Esta especificación de API se puede observar en el Anexo 8.6.

En caso de que un usuario necesite créditos para acceder a la plataforma, se ofrece a los usuarios con perfil administrador un método para poder agregarlos. En el alcance de este TFM no está la plataforma y la integración con un sistema de cobro automático como Stripe [39], debido al coste asociado de este. En la llamada, el administrador incluirá el usuario al que van dirigidos los créditos, los créditos a agregar y la información del pago, los cuales irán dentro de su estructura de datos correspondiente. Este método devolverá un 200 OK en caso de que todo haya ido de forma correcta, un 403 Forbidden en caso de que el usuario que esté intentando agregar créditos no sea administrador, y un 404 Not Found en caso de que el usuario al que se le esté intentando agregar los créditos no exista. Este método se puede ver en el Anexo 8.7.

Finalmente, se ha habilitado el acceso para que otros módulos puedan sustraer créditos de un usuario cuando realice una inferencia y puedan devolver créditos en caso de que estén informando al sistema del valor verdadero, incluyendo también la trazabilidad de cada una de estas operaciones, para poder tener no solo un histórico de las compras de créditos sino también de todos los eventos relacionados.

4.5.4. Módulo de gestión de modelos

Una vez se han creado los módulos de soporte del sistema, se crea una de las piedras angulares de la plataforma: el módulo de gestión de modelos. Este dispondrá de todos los elementos necesarios para poder agregar un modelo al sistema, ver los modelos disponibles

y eliminar un modelo anticuado. Además, aquí entra en consideración la integración con el sistema gestor de experimentos de MLFlow para utilizar los artefactos almacenados dentro del Bucket de S3 de este sistema y referenciarlos así a la plataforma principal. Un diagrama de este módulo se puede ver en la Figura 4.18.

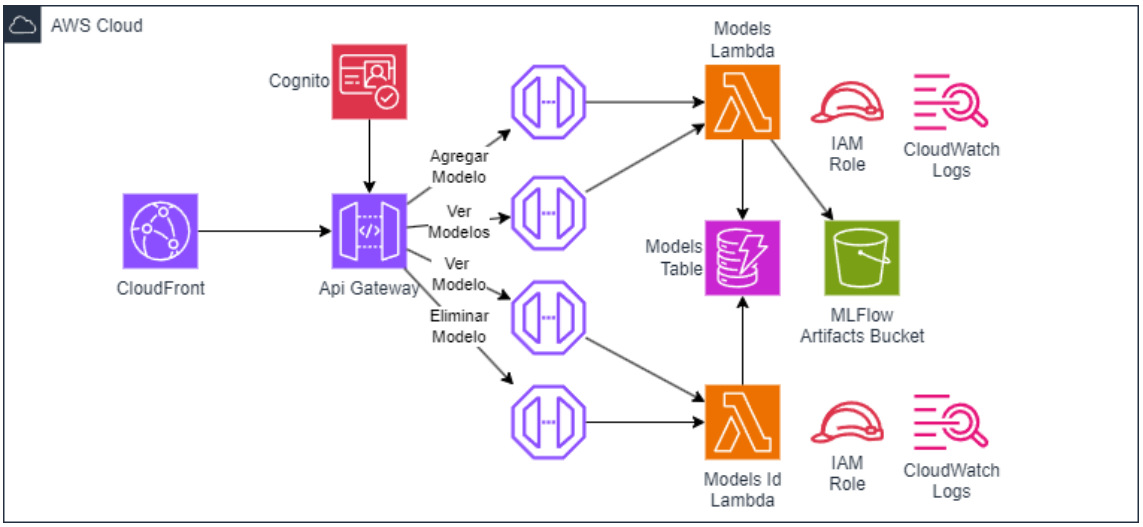

Figura 4.18. Diagrama de arquitectura del módulo de gestión de modelos

De igual manera, se ha seguido una arquitectura Serverless, utilizando DynamoDB para el almacenamiento de datos y AWS Lambda para el procesamiento de la información. Se han incluido los diferentes métodos dentro del mismo endpoint de AWS API Gateway.

Cuando se genere un modelo y sea necesario su puesta en producción, el primer método es el encargado de esta misión. Este método recibirá por parámetros la información del modelo, incluyendo el nombre, la descripción, el algoritmo utilizado y demás información relacionada con este. También se recibe la ruta de S3 en la cual se aloja el objeto pickle para poder cargarlo y efectuar la inferencia. Este método devolverá un 201 Created en caso de que el modelo haya sido creado de forma correcta, un 403 Forbidden en caso de que el usuario que esté intentando agregar el modelo no sea administrador y un 500 Internal Server Error en caso de que haya ocurrido algún error en el proceso de inclusión del modelo. La especificación de la API se puede ver en el Anexo 8.9.

El segundo método permite consultar los modelos disponibles como una lista, ofreciendo todas las características para invocarlos y compararlos entre sí, para elegir el mejor o el que más se adapte a las necesidades, ya sea por precio o por características del modelo. Este devolverá un 200 OK junto con la lista de modelos y un 404 Not Found en caso de que no exista ningún modelo para mostrar. Esta especificación de la API se puede ver en el Anexo 8.8.

De igual manera, es posible obtener únicamente la información de un único modelo, obteniendo así todas sus características. Este devuelve una respuesta 200 OK en caso de que se pueda devolver y todo haya salido correctamente, y un 404 Not Found en caso de que no exista el modelo al que se está intentando acceder. Esta especificación de la API se encuentra en el Anexo 8.10.

Finalmente, se propone un método para eliminar los modelos que hayan tenido unas métricas reales muy pobres, que se han reentrenado obteniendo unas métricas mejores con los nuevos registros o que se han decidido eliminar por cualquier otra razón. Este método devuelve un 200 OK en caso de que se haya eliminado el modelo de forma correcta, un 403 Forbidden en caso de que el usuario que esté intentando acceder no tenga rol de administrador y un 404 Not Found en caso de que el modelo no exista, como se puede ver en el Anexo 8.11.

4.5.5. Módulo de gestión de ejecuciones

Finalmente, llegamos al módulo que será usado en la mayoría de consumos de este sistema de servicio de modelos de aprendizaje automático, el módulo de gestión de ejecuciones, ya que este tiene como misión principal llevar a cabo la inferencia y predicción, aportando así valor al negocio. Este módulo no solo contempla este método, sino también los métodos necesarios para poder consultar el histórico de las ejecuciones que un usuario ha hecho, obtener una sola ejecución o poder enviar al sistema el valor real de una determinada ejecución. Además, en este módulo se incluye el servicio para poder, de forma periódica, calcular las métricas de los modelos y avisar al usuario administrador que es necesario reentrenarlo. Un diagrama conceptual de esta arquitectura se puede observar en la Figura 4.19.

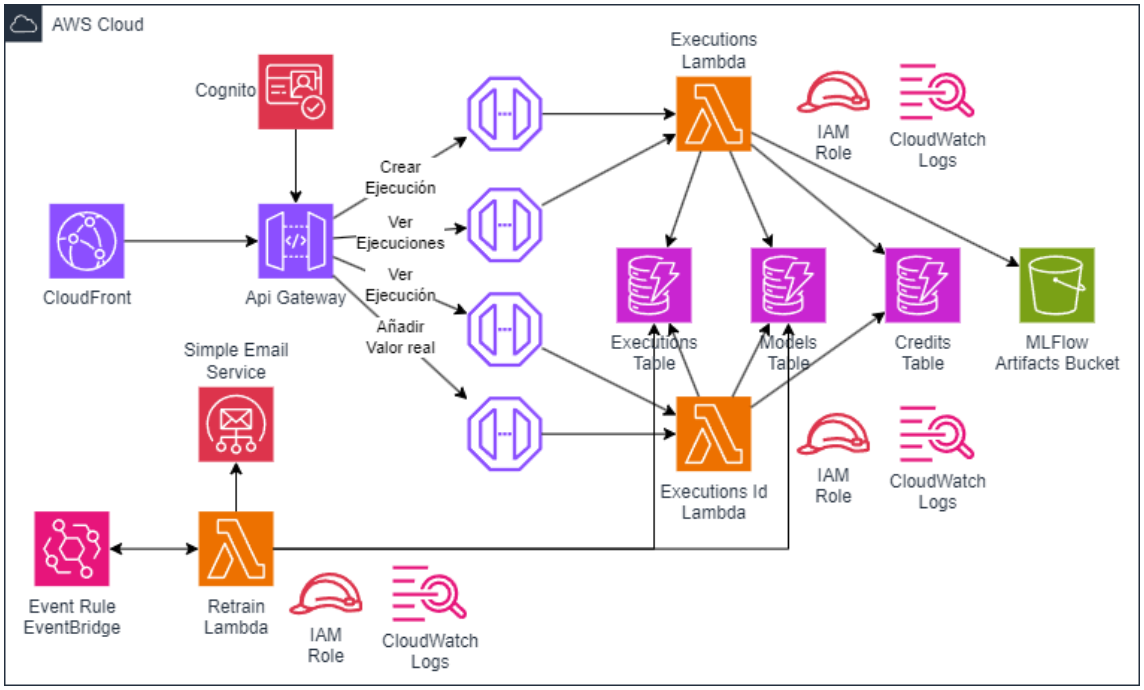

Figura 4.19. Diagrama de arquitectura del módulo de gestión de ejecuciones

En primer lugar, se sigue la misma lógica de todo el proyecto, realizándose sobre un entorno Serverless utilizando servicios como AWS Lambda, DynamoDB, S3 y API Gateway. Además, en este módulo se requiere de un cronjob de AWS EventBridge para poder realizar un evento todos los días en la madrugada, así como el uso de AWS SES (Simple Email Service) para enviar el correo electrónico de forma sencilla.

El primer método es el de realizar una inferencia. En este, lo primero es importar todas las librerías necesarias para realizar la inferencia, las cuales han sido previamente cargadas utilizando una Layer de AWS Lambda, que más tarde se comentará debido a la dificultad que ha entrañado dentro del proyecto. Este método, en primer lugar, comprueba que el usuario tiene créditos disponibles para poder efectuar la inferencia y prepara el objeto de historial correspondiente y la sustracción de créditos. A continuación, se descarga el modelo de S3 en la carpeta temporal para poder cargarlo en tiempo de ejecución. Con el modelo cargado, se serializan dentro de un array de numpy los datos de entrada y se realiza finalmente la predicción, guardándose además tanto en la tabla de créditos como en la de ejecuciones. Finalmente, el sistema devolverá un 200 OK junto con el identificador de la ejecución y la predicción hecha. Además, en caso de que no se encuentre el modelo, se devolverá 404 Not Found y, en caso de que no disponga de créditos suficientes para la inferencia, se devolverá 403 Insufficient Credits. La especificación de esta API se puede ver en el Anexo 8.12.

También se proporciona al usuario un método para poder visualizar todas las ejecuciones que ha realizado para ese modelo en particular, listándolas y obteniendo tanto los datos de entrada como el valor de predicción y el valor real en caso de que se haya agregado, en cuyo caso se devuelve un 200 OK. Además, es importante mencionar que para poder realizar esta consulta es necesario crear un índice (Global Secondary Index [40]) por usuario, de forma que se puedan hacer consultas sobre este valor en lugar de un escaneo de la tabla completa. Este método se puede ver en su tabla de API en el Anexo 8.13.

De igual manera que con los módulos anteriores, también se proporciona un método para poder obtener una única predicción y evitar así consultar la tabla entera para encontrar el valor deseado. Este devolverá 200 OK en caso de obtenerse de manera correcta junto con la información de la ejecución y 404 Not Found en caso de que no haya podido encontrarse la ejecución. Esto puede verse en el Anexo 8.14.

Finalmente, se establece un método para obtener la información real de los usuarios que usen la plataforma y con estos poder tanto recompensar en forma de créditos a aquellos que incluyan un valor al ecosistema colaborativo como poder medir de forma empírica la deriva del modelo y crear los mecanismos necesarios para poder reentrenarlo. Este método comprueba que el usuario que ha realizado la predicción es el que está incluyendo el valor real, genera el objeto de devolución de créditos y actualiza tanto la tabla de ejecuciones como la de créditos con estos datos. Devuelve 200 OK en caso de que todo haya ido correctamente y 404 Not Found en caso de que no se haya encontrado la ejecución o el usuario no corresponda con el creador. Esto se puede encontrar en la especificación de API del Anexo 8.15.

En este módulo también se ha integrado la ejecución para comprobar la deriva y el reentrenamiento. A la hora de diseñar este sistema se ha planteado un disparador que esté monitorizando la tabla de ejecuciones y calcule las métricas utilizando una lambda que se ejecute una vez por ejecución. Aunque debido a que el reentrenamiento no se realiza de manera automática y un operario tiene que decidir reentrenar el modelo, este servicio se ha establecido para que se genere un evento cada 24 horas y se realice este cálculo. En este servicio, primeramente, se obtienen los modelos y se va iterando por cada uno de ellos, se obtienen las ejecuciones de cada uno que tengan valor real y se calculan las métricas. Finalmente, se

construye un mensaje incluyendo una lista de modelos, las métricas de cada uno con el valor actual, el valor de la predicción, así como la variación porcentual de uno a otro. Con este mensaje se envía un correo de forma automatizada al administrador para poder llevar a cabo alguna acción con respecto al modelo.

Creación AWS Lambda Layer

El proceso de creación de la Layer encargada de realizar la predicción no ha sido una tarea sencilla como al principio se esperaba, ya que existe una limitación de 250MB en el entorno de ejecución de las Lambda provistas por AWS [41] y una layer con pandas, MLFlow, Scikit-learn, numpy y demás librerías comunes en este stack tecnológico superaba con creces este requerimiento. Es por ello que se ha barajado crear un entorno de ejecución personalizado con las librerías incluidas dentro de una imagen Docker o reducir el tamaño de la lambda.

Finalmente, tras investigar, se ha decidido intentar reducir el tamaño de la lambda layer. Para ello se ha basado esta creación en el repositorio aws-lambda-layer de Imperva en GitHub [42]. En este, lo que se hace es eliminar ciertos archivos de configuración, pruebas y datasets. De esta forma se puede llegar a reducir la layer por debajo del umbral para incluirla dentro del entorno de ejecución. Este script se puede ver en el Listing 4.2.

Listing 4.2: Script para reducción de tamaño de AWS Lambda Layer [42]

```
find /tmp/layer/python/ -type f -name "*.so" | xargs -r
rm -rf /tmp/layer/python/sklearn/datasets
find /tmp/layer/python/ -path "**/*.dist-info/*" -delete
find /tmp/layer/python/ -name "*fortran*.so" -delete
find /tmp/layer/python/ -name "*.pyc" -delete
find /tmp/layer/python/ -name "*.dat" -delete
find /tmp/layer/python/ -path "**/__pycache__/*" -delete
find /tmp/layer/python/ -path "**/tests/*" -delete
find /tmp/layer/python/ -path "**/tests/*" -delete
find /tmp/layer/python/ -path "**/test/*" -delete
```

4.5.6. Módulo de estadísticas

El último módulo es otro módulo auxiliar para obtener ciertas estadísticas de los modelos, las ejecuciones y los usuarios. Este consta de un endpoint de API y una lambda que procesa la información. Esta debe tener acceso a todas las tablas para obtener de ellas los elementos y poder realizar operaciones con ellos. El diagrama de este servicio se puede ver en la Figura 4.20.

A continuación, se muestra una lista de las estadísticas obtenidas a partir de hacer una llamada de API a este endpoint de estadísticas, tanto las estadísticas generales como las relacionadas con los usuarios, el cual se puede ver en el Anexo 8.16.

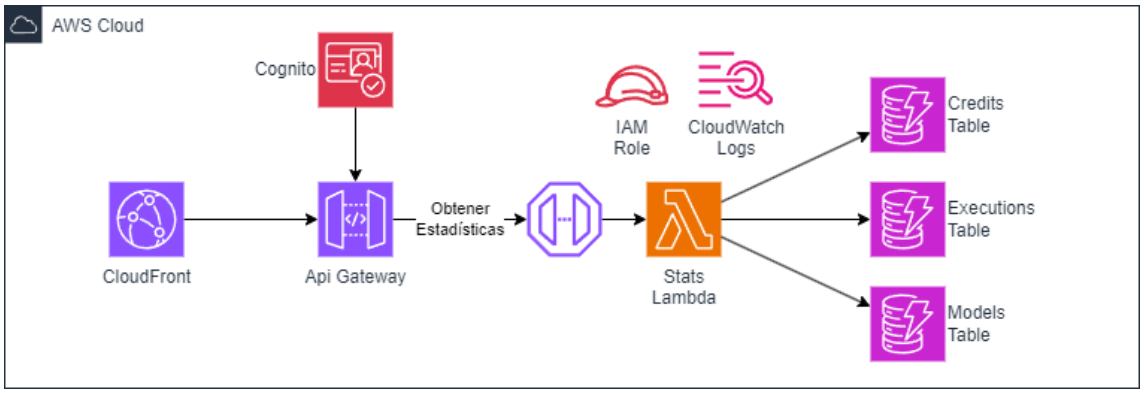

Figura 4.20. Diagrama de arquitectura del módulo de estadísticas

Estadísticas Generales

1. **Número total de ejecuciones:** Número total de ejecuciones realizadas.
2. **Total de créditos consumidos en ejecuciones:** Suma total de los créditos consumidos en todas las ejecuciones.
3. **Promedio de créditos consumidos por ejecución:** Promedio de créditos consumidos en cada ejecución.
4. **Desviación estándar de los créditos consumidos por ejecución:** Variabilidad de los créditos consumidos en las ejecuciones.
5. **Ejecución más reciente**
6. **Ejecución más antigua**
7. **Mediana de créditos consumidos por ejecución:** Valor central de los créditos consumidos en las ejecuciones.
8. **Máximo de créditos consumidos en una ejecución:** Máximo número de créditos consumidos en una sola ejecución.
9. **Mínimo de créditos consumidos en una ejecución:** Mínimo número de créditos consumidos en una sola ejecución.
10. **Total de créditos añadidos:** Suma total de los créditos añadidos a las cuentas.
11. **Promedio de créditos añadidos por transacción:** Promedio de créditos añadidos por cada transacción.
12. **Desviación estándar de los créditos añadidos por transacción:** Variabilidad de los créditos añadidos en las transacciones.

Estadísticas de Usuario

1. **Créditos totales disponibles:** Número total de créditos disponibles en la cuenta del usuario.

2. **Número total de ejecuciones del usuario:** Número total de ejecuciones realizadas por el usuario.
3. **Promedio de créditos consumidos por ejecución del usuario:** Promedio de créditos consumidos en cada ejecución del usuario.
4. **Desviación estándar de los créditos consumidos por ejecución del usuario:** Variabilidad de los créditos consumidos en las ejecuciones del usuario.
5. **Total de créditos consumidos en ejecuciones del usuario:** Suma total de los créditos consumidos en las ejecuciones del usuario.
6. **Mediana de créditos consumidos por ejecución del usuario:** Valor central de los créditos consumidos en las ejecuciones del usuario.
7. **Máximo de créditos consumidos en una ejecución del usuario:** Máximo número de créditos consumidos en una sola ejecución del usuario.
8. **Mínimo de créditos consumidos en una ejecución del usuario:** Mínimo número de créditos consumidos en una sola ejecución del usuario.
9. **Total de créditos añadidos por el usuario:** Suma total de los créditos añadidos a la cuenta del usuario.
10. **Promedio de créditos añadidos por transacción del usuario:** Promedio de créditos añadidos por cada transacción del usuario.
11. **Desviación estándar de los créditos añadidos por transacción del usuario:** Variabilidad de los créditos añadidos en las transacciones del usuario.
12. **Modelo más utilizado por el usuario**

4.6 Presupuesto

En esta sección se expone el presupuesto, el cual supone la evaluación económica total del proyecto, siendo un aspecto crucial para entender la viabilidad y la gestión de los recursos. Es fundamental considerar no solo los costes materiales o de software, sino también el valor del tiempo dedicado al proyecto.

Se presenta un resumen detallado del presupuesto, que incluye tanto los costes materiales como las horas de trabajo, está se puede ver en la Tabla 4.4. En esta están reflejados todos los costes, incluso aquellos que han sido nulos.

4.7 Viabilidad

En esta sección se realizará un breve estudio de viabilidad para establecer una base clara y permitir ajustes en las tecnologías empleadas en caso de que el planteamiento actual no resulte viable.

En este desarrollo, se consideran dos tipos de costes: el coste fijo, que incluye el desarrollo, mantenimiento y despliegue, y el coste variable asociado al funcionamiento de los diferentes módulos del sistema. Durante todo el proceso del TFM, el costo por infraestructura será nulo, ya que se dispone de la capa gratuita de AWS durante 12 meses. Por lo tanto, el único gasto

Tipo de coste	Valor	Comentarios
Horas de trabajo en el proyecto	140 horas	Horas aproximadas dedicadas al proyecto. Incluye el tiempo dedicado a la investigación, desarrollo, pruebas, documentación y exposición oral.
Horas de trabajo de colaboradores	10 horas	Horas aproximadas dedicadas por el tutor para tutorías y seguimiento
Ordenador de sobremesa	€1.000	Valor aproximado en el mercado del ordenador de sobremesa utilizado para el desarrollo y pruebas.
Ordenador portátil	€1.200	Valor aproximado en el mercado del ordenador portátil utilizado para movilidad y algunas tareas de desarrollo.
Monitores	€600	Valor aproximado en el mercado de los monitores utilizados.
Otros periféricos	€150	Valor aproximado en el mercado de los periféricos utilizados.
Cuenta AWS free tier	€0	Utilización de los servicios gratuitos proporcionados por AWS.
Visual Studio Code Insiders	€0	IDE utilizado para el desarrollo, gratuito.
Overleaf	€0	Herramienta online utilizada para la redacción de la memoria, cuenta gratuita.
PowerPoint	€0	Acceso a PowerPoint para la preparación de la defensa, proporcionado por la institución educativa.
Materiales empleados	€0	No se han empleado materiales adicionales como sensores o material de laboratorio.

Tabla 4.4. Costes del proyecto

a considerar es el de los recursos de desarrollo, estimado en 2.950 €, redondeando a 3.000 € para cubrir posibles imprevistos, costos extraordinarios de AWS o material necesario.

La factura de AWS puede variar significativamente debido a factores ocultos y fluctuaciones en el uso. Sin embargo, para este TFM, todo se realizará sobre la capa gratuita de AWS. Aun así, se hará una aproximación de costes e ingresos variables. Para el desarrollo de los experimentos, se utilizará una instancia EC2 t2.micro, con un costo aproximado de 0,011 €/h, lo que equivale a alrededor de 10 € al mes, incluyendo un extra de 2 € por la transferencia de datos.

Para la infraestructura de inferencia se consideran los siguientes componentes: ejecuciones de AWS Lambda, donde cada ejecución consume 2048 MB de memoria, con un costo aproximado de 0,01 € por cada 1.000 ejecuciones. Además, se tienen los accesos a S3 y DynamoDB. DynamoDB costará 0,03 € por cada 1.000 ejecuciones, mientras que S3 tendrá un costo de 0,01 € por cada 1.000 ejecuciones por la transferencia de datos y 0,005 € por cada 1.000 peticiones. Esto da un total de 0,055 € por cada 1.000 ejecuciones.

El sistema propuesto, aunque configurable, trata cada ejecución como el uso de 5 créditos y, si se proporciona el valor real, se devuelven 2 créditos.

Una de las principales bondades del enfoque ServerLess es su escalabilidad y pago por uso, lo que significa que si el volumen de ejecuciones es bajo, los costos también serán bajos. Esto facilita enormemente la creación de productos mínimos viables o pruebas de concepto sin necesidad de grandes inversiones iniciales en infraestructura. Así, se puede experimentar y ajustar el sistema antes de comprometerse con un despliegue a gran escala.

Para calcular la viabilidad, se deben considerar tanto los costes fijos como los costes variables. Los costes fijos ascienden a 3.000 € y los costes variables incluyen la instancia EC2 (10 € al mes), un cargo extra por transferencia de datos (2 € al mes), ejecuciones Lambda (0,01 € por cada 1.000 ejecuciones), DynamoDB (0,03 € por cada 1.000 ejecuciones) y S3 (0,015 € por cada 1.000 ejecuciones).

Los ingresos variables dependen del esquema de créditos. Si cada ejecución cuesta 5 créditos y se devuelven 2, en el caso extremo de que todas las ejecuciones sean subsanadas eficientemente, por lo que, el ingreso neto por ejecución es el equivalente a 3 créditos. Asumiendo que 1 crédito equivale a 0,01 €, cada ejecución genera 0,03 €.

Para encontrar el punto de equilibrio, se igualan los ingresos y los costos variables. Los ingresos por 1.000 ejecuciones son 30 € y los costes son 0,055 €. Dado que los ingresos superan ampliamente los costos variables, se deben cubrir los costes fijos de 3.000 €.

El número de ejecuciones necesario para alcanzar el punto de equilibrio es de aproximadamente 100.180 ejecuciones. Este análisis demuestra la viabilidad económica del proyecto siempre y cuando se realicen al menos 100.180 ejecuciones, tras lo cual comenzará a generarse beneficio neto.

$$\frac{3{,}000\,€}{30\,€/1{,}000\,\text{ejecuciones} - 0{,}055\,€/1{,}000\,\text{ejecuciones}} \approx 100{,}180\,\text{ejecuciones}$$

4.8 Resultados del proyecto

La forma óptima de determinar si el proyecto ha tenido éxito es comprobar los objetivos generales y los objetivos específicos marcados al inicio del proyecto, evaluando si se han conseguido y, en caso contrario, conocer las razones por las que no se ha logrado. También se debe señalar el plan de pruebas realizado para comprobar estos de forma empírica.

El primer objetivo general (OG.1) es desarrollar un modelo predictivo para abordar algún punto de dolor de la industria hotelera. Para ello, primero se ha estudiado la industria hotelera y su contexto, correspondiente al primer objetivo específico (OE.1), conociendo el porcentaje que representa la industria sobre el PIB, los retos que enfrenta con el cambio del modelo de consumo y el cambio tecnológico global. De esta forma, se ha podido determinar y entender el punto de dolor del impacto de las cancelaciones. Este objetivo se puede comprobar revisando el Capítulo 2.

Con este punto crítico identificado, se llevó a cabo una recopilación de una serie de datos

públicos relacionados con la demanda hotelera, así como su procesamiento y análisis para dejarlos preparados para la etapa posterior, pudiendo ser revisado en la Sección 4.4 y en las Subsecciones 4.4.2, 4.4.3 y 4.4.4, cumpliendo así el cuarto objetivo específico (OE.4). Posteriormente, se realizó el entrenamiento de diferentes modelos de aprendizaje automático, correspondiente al objetivo específico cinco (OE.5), probando un total de 11 modelos distintos y optimizando algunos de ellos mediante una búsqueda bayesiana. Finalmente, estos modelos fueron comparados utilizando un sistema gestor de modelos de aprendizaje automático en la nube, eligiendo así 2 modelos para llevar a producción en casos de uso diferentes, cumpliendo el objetivo específico seis (OE.6). Estos últimos dos objetivos pueden ser revisados en la Sección 4.4 y en la Subsección 4.4.5.

También se ha cumplido el segundo objetivo general (OG.2), que consistía en desplegar un sistema de gestión de modelos de aprendizaje automático en un entorno Cloud. De esta forma, también se han cumplido los objetivos específicos dos (OE.2) y tres (OE.3). El objetivo dos se refiere al uso de la tecnología Cloud como eje fundamental del trabajo, eligiendo AWS como proveedor debido a su coste asociado con la capa gratuita, tanto para el sistema gestor de modelos de aprendizaje automático como para el sistema de servicio de los modelos de aprendizaje automático. Por otro lado, el objetivo específico tres se ha cumplido al desplegar una instancia de cómputo en la que se ha montado de manera sencilla y automática el sistema gestor de MLFlow para registrar experimentos y diferentes ejecuciones. Este objetivo puede ser revisado en la Sección 4.3.

Finalmente, todo este trabajo previo culmina en la consecución del tercer objetivo general (OG.3), que es el desarrollo de un producto colaborativo para los diferentes agentes de la industria hotelera. Este producto contribuye a alcanzar el objetivo de conocer la industria hotelera (OE.1) al descubrir que no existe una gran colaboración entre las cadenas hoteleras. Del mismo modo, se consigue el objetivo de utilizar la tecnología Cloud (OE.2) al igual que en el sistema de gestión de modelos de aprendizaje automático. Este sistema se ha desarrollado también en AWS, aunque con una arquitectura mucho más serverless.

Este producto contiene diferentes módulos para interactuar con cada una de las partes y cumplir cada uno de los objetivos específicos. En primer lugar, interactuar de forma sencilla con los modelos, consiguiendo así el objetivo específico número siete (OE.7), desde la creación y manejo de usuarios, pasando por el consumo de créditos hasta la generación de estadísticas, y llegando finalmente a realizar la propia inferencia. Además, una vez ofrecido todo el sistema de inferencia, se fomenta la colaboración entre los agentes mediante incentivos y recompensas a los usuarios, quienes colaboran con el sistema para mejorarlo y crear un ecosistema mejor, cumpliendo así el objetivo específico número ocho (OE.8). Finalmente, gracias a todas estas colaboraciones, se puede medir y monitorear la deriva de los modelos y plantear su reentrenamiento de forma periódica, cumpliendo así el objetivo específico número nueve (OE.9). Este último objetivo puede ser revisado de manera conceptual en la Sección 4.5 y de manera empírica mediante el plan de pruebas expuesto en esta sección, contenido en las Figuras 4.21 a 4.35, así como una serie de pruebas adicionales en el Anexo 8.2, contenidas en las Figuras 8.1 a 8.12.

POST {{domain}}/users Send

Body raw JSON Beautify

```
{
    "username": "oscarjprietom@gmail.com",
    "password": "Abc1234!"
}
```

Body 200 OK 397 ms 650 B Save as example

Pretty JSON

```
{
    "CodeDeliveryDetails": {
        "AttributeName": "email",
        "DeliveryMedium": "EMAIL",
        "Destination": "o***@g***"
    },
    "UserConfirmed": false,
    "UserSub": "9245f424-70e1-70ee-624f-ef3b6a3f60e5"
}
```

Figura 4.21. Prueba crear usuario

N no-reply@verificationemail.com
Para: Óscar Javier Prieto Martínez

Este mensaje está en Inglés

The verification code to your new account is 284241

PUT {{domain}}/users/oscarjprietom@gmail.com/confirm Send

Body raw JSON Beautify

```
{
    "confirmationCode": "724905"
}
```

Body 200 OK 1053 ms 479 B Save as example

Pretty JSON

```
{}
```

Figura 4.22. Prueba confirmar de usuario

POST {{domain}}/users/oscarjprietom@gmail.com/forgotPassword Send

Body none

This request does not have a body

Body 200 OK 287 ms 579 B Save as example

Pretty JSON

```
{
    "CodeDeliveryDetails": {
        "AttributeName": "email",
        "DeliveryMedium": "EMAIL",
        "Destination": "o***@g***"
    }
}
```

PUT {{domain}}/users/oscarjprietom@gmail.com/forgotPassword/confirm Send

Body raw JSON Beautify

```
{
    "confirmationCode": "223039",
    "password": "Test1234!"
}
```

Body 200 OK 664 ms 479 B Save as example

Pretty JSON

```
{}
```

Figura 4.23. Prueba restablecer contraseña

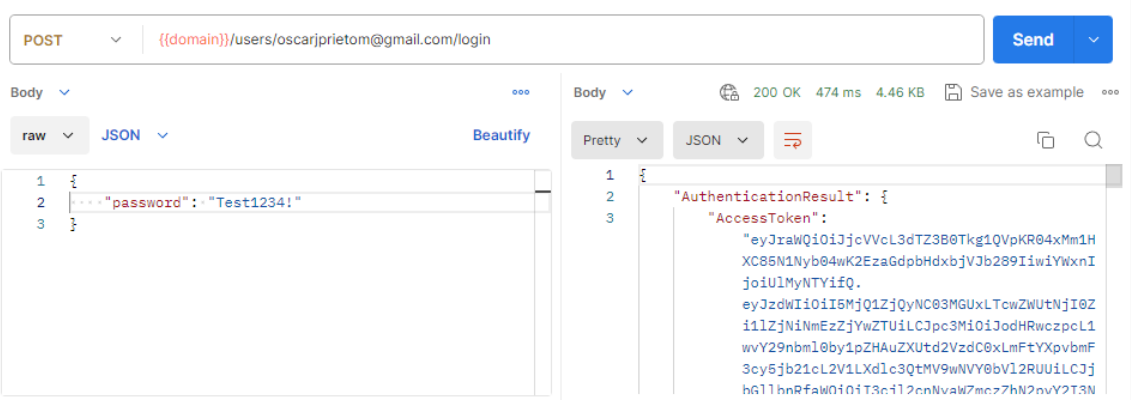

Figura 4.24. Prueba iniciar sesión

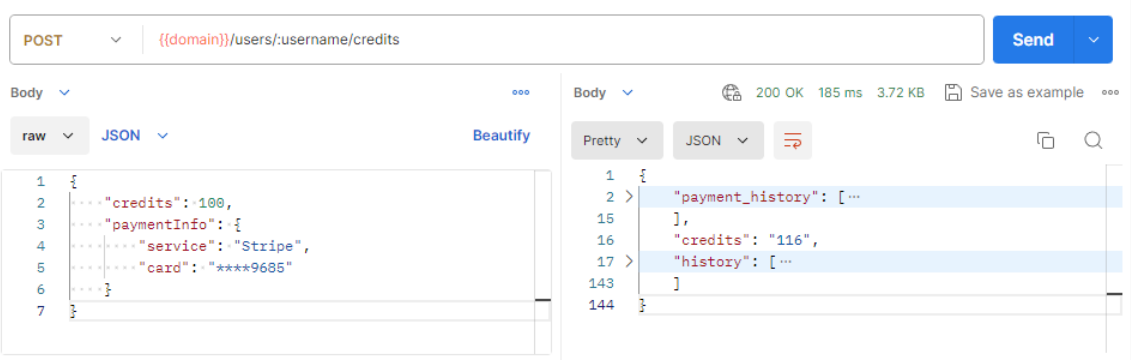

Figura 4.25. Prueba agregar créditos

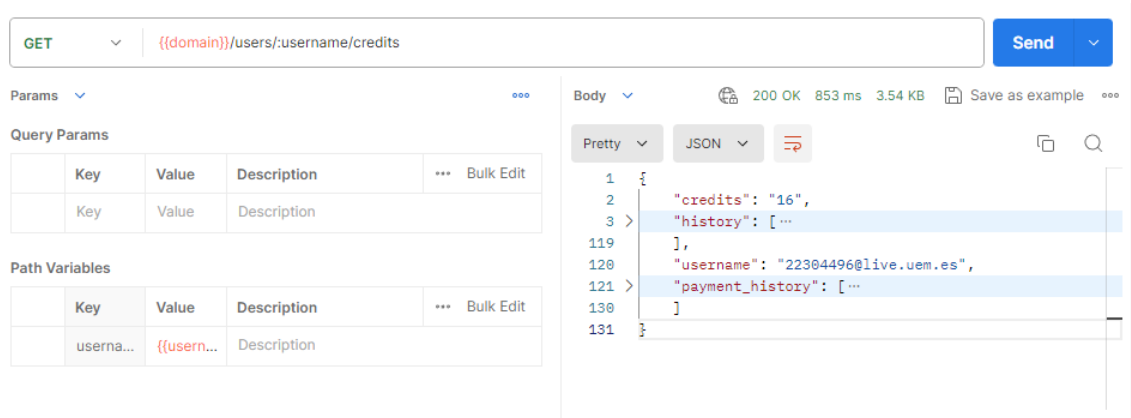

Figura 4.26. Prueba obtener créditos

POST {{domain}}/models Send

Body raw JSON Beautify


```
{
    "name": "Predicción de cancelaciones",
    "description": "",
    "algorithm": "DecisionTreeClassifier",
    "s3_key": "460891820793183679/
        86a8d4f914bc4c8a89f7ac971a6d8175/artifacts/
        model/model.pkl",
    "metrics": {
        "accuracy": 0.8507569687513178,
        "precision": 0.8008257085471993,
        "recall": 0.8036054193259433,
        "f1_score": 0.8022131559827865,
        "roc_auc": 0.8414252235311898
    },
    "consumed_credits": 2,
    "returned_credits": 1,
    "input_schema": {
        "$schema": "http://json-schema.org/draft-07/
```


Body 201 Created 276 ms 570 B Save as example

Pretty JSON


```
{
    "message": "Model created successfully",
    "id": "d9d7dfde-40c5-49c0-87af-e53d1a1b4851"
}
```


Figura 4.27. Prueba crear modelo

GET {{domain}}/models Send

Body none

This request does not have a body

Body 200 OK 406 ms 13.93 KB Save as example

Pretty JSON


```
[
    {
        "run_id": "86a8d4f914bc4c8a89f7ac971a6d8175",
        "created_at": "2024-07-14T16:50:23.557195",
        "consumed_credits": 2.0,
        "experiment_id": "460891820793183679",
        "name": "Predicción de cancelaciones",
        "returned_credits": 1.0,
        "updated_at": "2024-07-14T16:50:23.557195",
        "input_schema": {
            "allOf": [
                {
```


Figura 4.28. Prueba listar modelos

GET {{domain}}/models/:modelId Send

Params

Query Params

	Key	Value	Description	Bulk Edit
	Key	Value	Description	

Path Variables

	Key	Value	Description	Bulk Edit
	modelId	049f94...	Description	

Body 200 OK 356 ms 7.19 KB Save as example

Pretty JSON


```
{
    "updated_at": "2024-07-08T13:35:28.934201",
    "input_schema": {...
    },
    "run_id": "3c80e7862c284c5b9931b4247e4bd51b",
    "created_at": "2024-07-08T13:35:28.934201",
    "algorithm":
        "RandomForestClassifier-BayesSearchCV",
    "consumed_credits": 5.0,
    "experiment_id": "460891820793183679",
    "metrics": {
        "accuracy": 0.8860962341331758,
```


Figura 4.29. Prueba obtener modelo

Figura 4.30. Prueba eliminar modelo

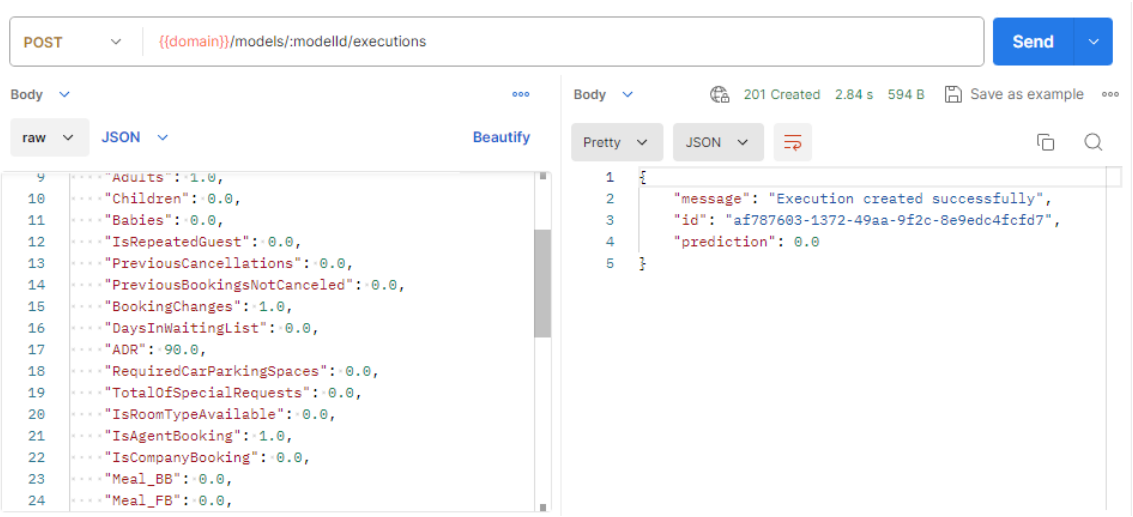

Figura 4.31. Prueba crear ejecución

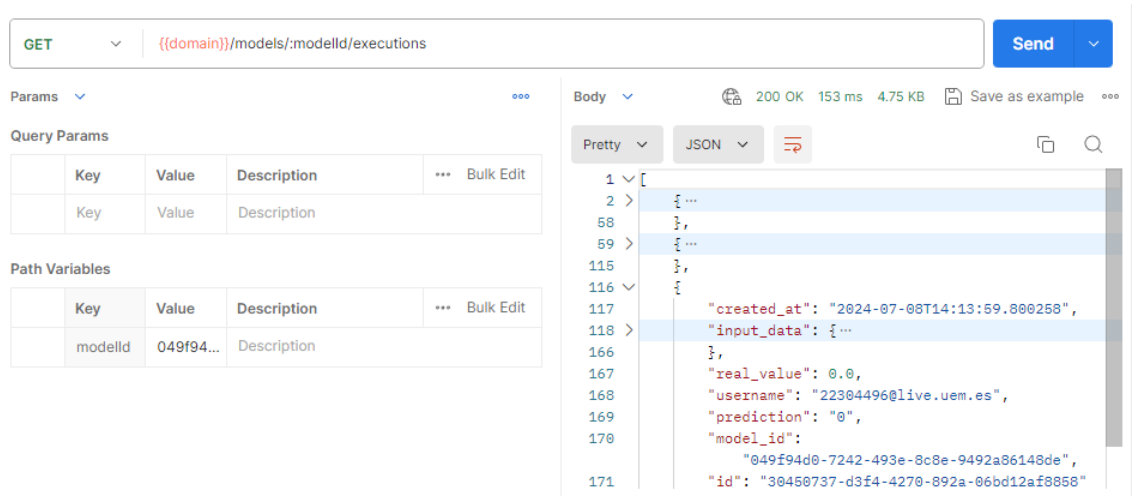

Figura 4.32. Prueba listar ejecuciones

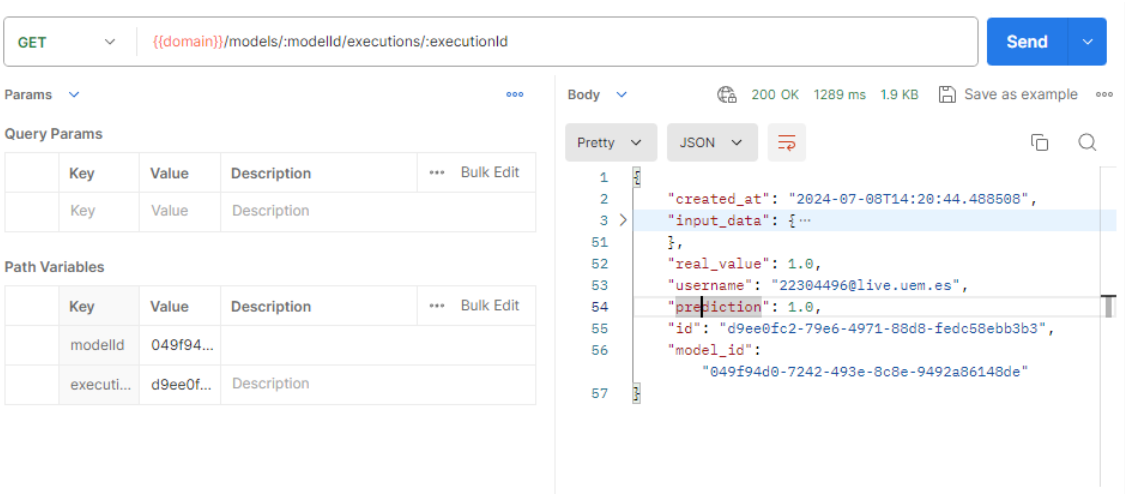

Figura 4.33. Prueba obtener ejecución

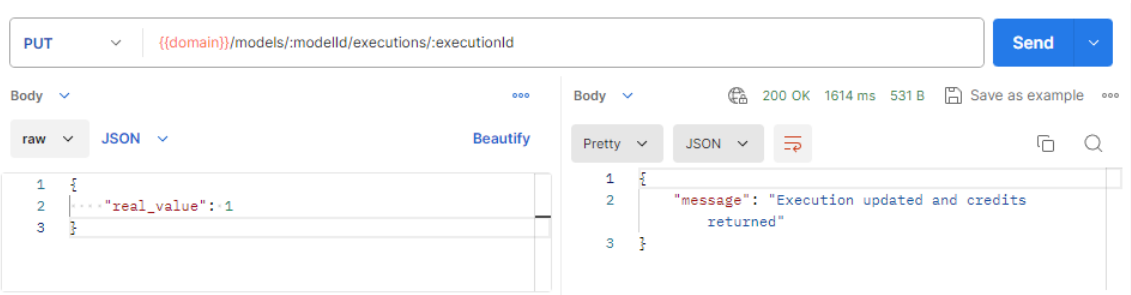

Figura 4.34. Prueba añadir valor verdadero

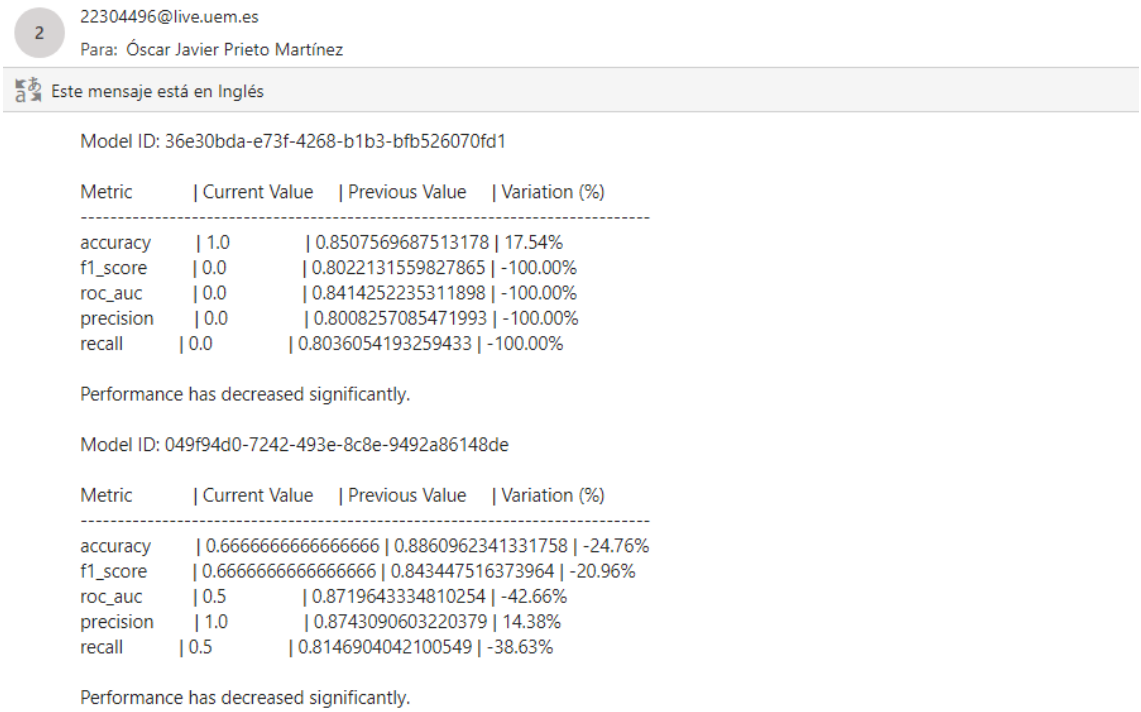

Figura 4.35. Prueba aviso de reentrenamiento

Capítulo 5. CONCLUSIONES

En este capítulo se exponen las conclusiones obtenidas en el trabajo, tanto las conclusiones propias del trabajo, incluyendo las limitaciones del estudio y los cambios respecto al planteamiento inicial, como las conclusiones personales al trabajar con esta tecnología y lo que ha supuesto realizar este trabajo a nivel personal.

5.1 Conclusiones del trabajo

Este Trabajo Fin de Máster ha sido un éxito rotundo al cumplir con los tres objetivos principales establecidos desde el inicio, como se detalla en la Sección 4.8. Primero, logrando establecer un sistema de gestión de experimentos de aprendizaje automático sobre una infraestructura en la nube. Posteriormente, completando todo el ciclo de vida de la creación de modelos de aprendizaje automático para resolver un problema real. Y finalmente, desarrollando un producto colaborativo que permite servir estos modelos de manera eficiente en un entorno en la nube.

Además de cumplir con los tres objetivos generales y los nueve específicos establecidos al principio, el trabajo ha demostrado la viabilidad y el impacto positivo del uso de tecnología avanzada para resolver problemas en la industria hotelera. La implementación exitosa de un sistema predictivo y colaborativo en la nube, junto con la optimización y despliegue de modelos de aprendizaje automático, ofrece una solución robusta y efectiva. Esta realización no solo valida la metodología empleada, sino que también impulsa el uso de tecnologías de aprendizaje automático, promoviendo la innovación y la mejora continua en la gestión hotelera.

5.2 Conclusiones personales

A título personal, este proyecto no solo me ha permitido aprender nuevas tecnologías o aplicar aquellas que ya conocía, sino que también ha supuesto la resolución de un problema desde cero. Este aspecto es fundamental para nosotros como ingenieros, ya que define nuestra capacidad de aplicar las técnicas y tecnologías aprendidas al servicio de la sociedad, mejorando así la vida de las personas.

En este proyecto he puesto en práctica todo lo aprendido desde el inicio de este máster, desde el uso de arquitecturas Cloud avanzadas, las cuales pueden ser desplegadas como código de forma eficiente, hasta la realización de todo el desarrollo basándome en buenas prácticas. Además, he aplicado mis conocimientos en procesamiento de datos y aprendizaje automático, lo que me ha permitido decidir cómo resolver problemas y optimizarlos de manera eficaz. Todo esto integrado en un producto capaz de satisfacer una necesidad de forma adecuada.

Este proyecto no solo marca el final de un ciclo académico, sino también el comienzo de una nueva etapa en el mundo profesional llena de retos y aprendizajes. Me siento emocionado y preparado para enfrentar el futuro, y estoy ilusionado y dispuesto a seguir creando soluciones que realmente hagan la diferencia en la vida de las personas.

Capítulo 6. FUTURAS LÍNEAS DE TRABAJO

En este capítulo, se exponen las futuras líneas de trabajo que se proponen para poder continuar el trabajo realizado, permitiendo así abarcar aspectos fuera del alcance del mismo y que son interesantes de desarrollar.

En los proyectos de software el alcance puede llegar a ser infinito, agregándole innumerables características, dificultando así establecer un alcance claro al inicio, teniendo que priorizar funcionalidades y recursos para poder conseguir el mejor producto posible en el tiempo acotado para ello, al igual que ocurre en el mundo profesional.

La primera necesidad para un trabajo futuro es conseguir más datos, ya sea mediante una colaboración directa con alguna cadena hotelera, a través de una empresa intermediaria o mediante cualquier método lícito. En este caso, el producto y el modelo están muy enfocados en un segmento de mercado en concreto y en un país en concreto, pero si se consiguieran datos suficientes y diversos, se podría crear un sistema mejor.

Por otro lado, el objetivo de crear todo el producto encapsulado dentro de una API REST es poder ser lo más portable posible e integrarse dentro de un gran número de sistemas. Así se podría mejorar la experiencia de usuarios menos especializados, proponiendo la creación de un front-end para la interacción o creando diversos plugins de sistemas gestores de reservas o ERPs especializados.

En este proyecto, debido a la realización del entrenamiento on-premise y por temas de coste, el reentrenamiento de los modelos se ha dejado a discreción de los administradores en base a la información proporcionada por el sistema, lo cual puede resultar poco escalable. Es por esto que se puede realizar esta acción directamente en el entorno en la nube, agregándole una aprobación del operario antes de la subida a producción, por ejemplo.

Además, se tiene constancia que las empresas cada vez más intentan no depender únicamente de un solo proveedor Cloud y evitar problemas como los originados en los sistemas Windows por el software de CloudStrike [43]. Es por esto que se propone un desarrollo de la misma infraestructura en un entorno MultiCloud, ampliando la complejidad del sistema pero mejorando la robustez de la solución.

En resumen, se ha recorrido gran parte del camino, pero a partir de este punto existen muchas oportunidades para innovar y mejorar, perfeccionando el trabajo actual y generando de forma iterativa nuevas líneas futuras, adaptándose cada vez a tecnologías más avanzadas. De esta forma, no solo se propone la creación de un ecosistema colaborativo en el reentrenamiento de los modelos, sino también el propio desarrollo de la solución en sí.

Capítulo 7. BIBLIOGRAFÍA

[1] *Exceltur — ICEX Invest in Spain*, Investinspain.org, 2023. dirección: https://www.investinspain.org/content/icex-invest/es/noticias-main/2024/exceltur.html.

[2] *Cuentas Nacionales Trimestrales de España*, INE, 2016. dirección: https://ine.es/dyngs/INEbase/es/operacion.htm?c=Estadistica_C&cid=1254736169169&menu=ultiDatos&idp=1254735576863.

[3] *Hotels continue to recover in 2023 as the industry begins a new era of operations*, Ahla.com, 2023. dirección: https://www.ahla.com/news/report-hotels-continue-recover-2023-industry-begins-new-era-operations.

[4] E. Lacalle, *¿A dónde se dirige la digitalización del sector hotelero?* Mews.com, abr. de 2023. dirección: https://www.mews.com/es/blog/digitalizacion-sector-hotelero.

[5] Sfaura, *Sostenibilidad en hoteles: así se adaptan los alojamientos y las OTAS a las nuevas demandas*, TecnoHotel, jul. de 2023. dirección: https://tecnohotelnews.com/2023/07/sostenibilidad-en-hoteles-medidas/.

[6] TH Redacción, *Ratios de cancelación, una pieza clave en la estrategia tarifaria de nuestro negocio*, TecnoHotel, oct. de 2023. dirección: https://tecnohotelnews.com/2023/10/ratios-de-cancelacion-2023-infografia/.

[7] C. Rentero, *Los hoteles ante el aumento de las cancelaciones, ¿qué hacer?* TecnoHotel, jul. de 2019. dirección: https://tecnohotelnews.com/2019/07/hoteles-cancelaciones-actuaciones/.

[8] C. Rentero, *Viajeros que reservan por si acaso y el drama de las cancelaciones para los hoteles*, Smart Travel News, mar. de 2019. dirección: https://www.smarttravel.news/viajeros-reservan-acaso-drama-las-cancelaciones-los-hoteles/.

[9] *Get started with the AWS Free Tier*, Amazon.com, 2024. dirección: https://docs.aws.amazon.com/whitepapers/latest/how-aws-pricing-works/get-started-with-the-aws-free-tier.html.

[10] *MLflow*, Mlflow.org, 2024. dirección: https://mlflow.org/.

[11] *What is Infrastructure as Code (IaC)?* Redhat.com, 2022. dirección: https://www.redhat.com/es/topics/automation/what-is-infrastructure-as-code-iac.

[12] *AWS CDK Python Reference — AWS Cloud Development Kit 2.144.0 Documentation*, Amazon.com, 2020. dirección: https://docs.aws.amazon.com/cdk/api/v2/python/.

[13] *Bootstrapping — AWS Cloud Development Kit (CDK) V2*, Amazon.com, 2022. dirección: https://docs.aws.amazon.com/cdk/v2/guide/bootstrapping.html.

[14] *sklearn.datasets.load*$_{wine^{-}scikit-learn1,0,0documentation}$, scikit-learn, 2024. dirección: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_wine.html.

[15] P. C. del Saz Orozco Huang, *Aprendizaje Automático*, Europaeducationgroup.es, 2020. dirección: https://campus.europaeducationgroup.es/courses/46137/files/10400749?module_item_id=1260476.

[16] *Centro de Computación Avanzada*, Universidad Europea, 2022. dirección: https://universidadeuropea.com/tu-experiencia/instalaciones/centro-de-computacion-avanzada/.

[17] N. Antonio, A. de Almeida y L. Nunes, «Hotel booking demand datasets,» *Data in Brief*, vol. 22, págs. 41-49, 2019, ISSN: 2352-3409. DOI: https://doi.org/10.1016/j.dib.2018.11.126. dirección: https://www.sciencedirect.com/science/article/pii/S2352340918315191.

[18] International Organization for Standardization, *ISO 3166-3:2013*, ISO, 2024. dirección: https://www.iso.org/standard/63547.html.

[19] *What is Feature Engineering?* Ibm.com, ene. de 2024. dirección: https://www.ibm.com/topics/feature-engineering.

[20] F. Data, *One-Hot Encoding — A Brief Explanation - Wojtek Fulmyk, Data Scientist - Medium*, jul. de 2023. dirección: https://medium.com/@WojtekFulmyk/one-hot-encoding-a-brief-explanation-8c5daec395e3.

[21] *What is Logistic Regression?* Ibm.com, ago. de 2021. dirección: https://www.ibm.com/topics/logistic-regression.

[22] *¿Qué es el algoritmo de K vecinos más cercanos?* Ibm.com, jul. de 2022. dirección: https://www.ibm.com/es-es/topics/knn.

[23] *What is a Decision Tree?* Ibm.com, nov. de 2021. dirección: https://www.ibm.com/topics/decision-trees.

[24] *What is Random Forest?* Ibm.com, oct. de 2021. dirección: https://www.ibm.com/topics/random-forest.

[25] *¿Qué es AdaBoost? Concepto y Definición*, GAMCO, SL, abr. de 2023. dirección: https://gamco.es/glosario/adaboost/.

[26] J. Hoare, *Gradient Boosting Explained - The Coolest Kid on The Machine Learning Block*, Displayr, jun. de 2017. dirección: https://www.displayr.com/gradient-boosting-the-coolest-kid-on-the-machine-learning-block.

[27] *XGBoost Documentation*, Readthedocs.io, 2022. dirección: https://xgboost.readthedocs.io/en/stable.

[28] *Welcome to LightGBM's Documentation!* Readthedocs.io, 2023. dirección: https://lightgbm.readthedocs.io/en/stable/.

[29] K. Thankachan, *What? When? How?: ExtraTrees Classifier - Towards Data Science*, Medium, ago. de 2022. dirección: https://towardsdatascience.com/what-when-how-extratrees-classifier-c939f905851c.

[30] GeeksforGeeks, *Voting Classifier*, GeeksforGeeks, oct. de 2023. dirección: https://www.geeksforgeeks.org/voting-classifier/.

[31] *Scikit-learn: Machine Learning in Python*, Scikit-learn.org, 2024. dirección: https://scikit-learn.org/stable/.

[32] *State-of-the-art Open-source Gradient Boosting Library with Categorical Features Support*, Catboost.ai, 2024. dirección: https://catboost.ai/.

[33] L. Breiman, «Bagging predictors,» *Machine Learning*, vol. 24, n.º 2, págs. 123-140, 1996, ISSN: 1573-0565. DOI: 10.1007/BF00058655. dirección: https://doi.org/10.1007/BF00058655.

[34] L. Breiman, «Random Forests,» *Machine Learning*, vol. 45, n.º 1, págs. 5-32, 2001, ISSN: 1573-0565. DOI: 10.1023/A:1010933404324. dirección: https://doi.org/10.1023/A:1010933404324.

[35] Y. Freund y R. E. Schapire, «A Short Introduction to Boosting,» en *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 1999. dirección: https://api.semanticscholar.org/CorpusID:9621074.

[36] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush y A. Gulin, *CatBoost: unbiased boosting with categorical features*, 2019. arXiv: 1706.09516 [cs.LG]. dirección: https://arxiv.org/abs/1706.09516.

[37] M. Gorodetski, *Hyperparameter Tuning Methods - Grid, Random or Bayesian Search?* Medium, ago. de 2021. dirección: https://towardsdatascience.com/bayesian-optimization-for-hyperparameter-tuning-how-and-why-655b0ee0b399.

[38] G. L. Team, *Hyperparameter Tuning with GridSearchCV*, Great Learning Blog: Free Resources What Matters to Shape Your Career, mayo de 2023. dirección: https://www.mygreatlearning.com/blog/gridsearchcv/.

[39] *Stripe*, Stripe.com, 2024. dirección: https://stripe.com/es.

[40] *Using Global Secondary Indexes in DynamoDB - Amazon DynamoDB*, Amazon.com, 2024. dirección: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GSI.html.

[41] *Cuotas de Lambda - AWS Lambda*, Amazon.com, 2024. dirección: https://docs.aws.amazon.com/es_es/lambda/latest/dg/gettingstarted-limits.html.

[42] Imperva, *GitHub - imperva/aws-lambda-layer*, GitHub, 2024. dirección: https://github.com/imperva/aws-lambda-layer.

[43] elEconomista.es, *Microsoft estima que el error de Crowdstrike afectó a 8.5 millones de dispositivos Windows*, elEconomista.es, jul. de 2024. dirección: https://www.eleconomista.es/tecnologia/noticias/12918808/07/24/microsoft-estima-que-el-error-de-crowdstrike-afecto-a-85-millones-de-dispositivos-windows.html.

Capítulo 8. ANEXOS

8.1 Definición de API

8.1.1. Módulo de usuarios

Tabla 8.1. Definición API creación de usuario

POST /users

Body application/json

```
{
    "username": String,
    "password": String
}
```

Response application/json

200 OK

```
{
    "CodeDeliveryDetails": {
        "AttributeName": String,
        "DeliveryMedium": String,
        "Destination": String
    },
    "UserConfirmed": boolean,
    "UserSub": String
}
```

400 Bad Request

```
{
    "__type": String,
    "message": String
}
```

Tabla 8.2. Definición API confirmación de usuario

PUT	/users/:username/confirm
Parameter	
username	Email/nombre de usuario
Body	application/json

```
{
    "confirmationCode": Number
}
```

Response	application/json
200	OK
400	Bad Request

```
{
    "__type": String,
    "message": String
}
```

Tabla 8.3. Definición API restablecemiento de contraseña

POST	/users/:username/forgotPassword
Parameter	
username	Email/nombre de usuario
Response	application/json
200	OK

```
{
    "CodeDeliveryDetails": {
        "AttributeName": String,
        "DeliveryMedium": String,
        "Destination": String
    }
}
```

400 Bad Request

```
{
    "__type": String,
    "message": String
}
```

Tabla 8.4. Definición API confirmación restablecemiento de contraseña

PUT	**/users/:username/forgotPassword/confirm**
Parameter	
username	Email/nombre de usuario
Body	application/json

```
{
    "confirmationCode": Number
}
```

Response	application/json
200	OK
400	Bad Request

```
{
    "__type": String,
    "message": String
}
```

Tabla 8.5. Definición API inicio de sesión

POST **/users/:username/login**

Parameter

username	Email/nombre de usuario

Body application/json

```
{
    "password": String
}
```

Response application/json

200 OK

```
{
    "AuthenticationResult": {
        "AccessToken": String,
        "ExpiresIn": Integer,
        "IdToken": String,
        "RefreshToken": String,
        "TokenType": "Bearer"
    },
    "ChallengeParameters": {}
}
```

400 Bad Request

```
{
    "__type": String,
    "message": String
}
```

8.1.2. Módulo de créditos

Tabla 8.6. Definición API obtener créditos del usuario

GET **/users/:username/credits**

Parameter

username	Email/nombre de usuario

Response application/json

200 OK

```
{
    "username": String,
    "credits": Number,
    "history": [
        {
            "payment_info": {
                "service": String,
                "card": String
            },
            "current_credits": Number,
            "added_credits": Number,
            "updated_credits": Number,
            "date": String
        }
    ]
}
```

403 Forbidden

```
{
    "message": "User not authorized"
}
```

404 Not Found

```
{
    "message": "User not found"
}
```

Tabla 8.7. Definición API agregar créditos al usuario

POST **/users/:username/credits**

Parameter

username	Email/nombre de usuario

Body application/json

```
{
    "credits": Number,
    "paymentInfo": {
        "service": String,
        "card": String
    }
}
```

Response application/json

200 OK

```
{
    "credits": Number,
    "history": [
        {
            "payment_info": Object,
            "current_credits": Number,
            "added_credits": Number,
            "updated_credits": Number,
            "date": String
        }
    ],
    "payment_history": [
        {
            "service": String,
            "card": String
        }
    ]
}
```

403 Forbidden

```
{
    "message": "User not authorized"
}
```

404 Not Found

```
{
    "message": "User not found"
}
```

8.1.3. Módulo de modelos

Tabla 8.8. Definición API listar modelos

GET **/models**

Response application/json

200 OK

```
[
    {
        "id": String,
        "created_at": String,
        "updated_at": String,
        "name": String,
        "description": String,
        "algorithm": String,
        "s3_key": String,
        "metrics": {
            "accuracy": String,
            "precision": String,
            "recall": String,
            "f1_score": String,
            "roc_auc": String
        },
        "consumed_credits": Number,
        "returned_credits": Number,
        "input_schema": JSONSchema,
        "experiment_id": String,
        "run_id": String
    }
]
```

500 Internal Server Error

```
{
    "message": String
}
```

Tabla 8.9. Definición API crear modelo

POST **/models**

Body application/json

```
{
    "name": String,
    "description": String,
    "algorithm": String,
    "s3_key": String,
    "metrics": {
        "accuracy": String,
        "precision": String,
        "recall": String,
        "f1_score": String,
        "roc_auc": String
    },
    "consumed_credits": Number,
    "returned_credits": Number,
    "input_schema": JSONSchema,
    "experiment_id": String,
    "run_id": String
}
```

Response application/json

201 Created

```
{
    "message": String,
    "id": String
}
```

403 Forbidden

```
{
    "message": String
}
```

500 Internal Server Error

```
{
    "message": String
}
```

Tabla 8.10. Definición API ver modelo

GET **/models/:modelId**

Parameter

modelId	ID del modelo

Response application/json

200 OK

```
{
    "id": String,
    "created_at": String,
    "updated_at": String,
    "name": String,
    "description": String,
    "algorithm": String,
    "s3_key": String,
    "metrics": {
        "accuracy": String,
        "precision": String,
        "recall": String,
        "f1_score": String,
        "roc_auc": String
    },
    "consumed_credits": Number,
    "returned_credits": Number,
    "input_schema": JSONSchema,
    "experiment_id": String,
    "run_id": String
}
```

404 Not Found

```
{
    "message": String
}
```

500 Internal Server Error

```
{
    "message": String
}
```

Tabla 8.11. Definición API eliminar modelo

DELETE	/models/:modelId
Parameter	
modelId	ID del modelo
Response	application/json
200	OK

```
{
    "message": String
}
```

404	Not Found

```
{
    "message": String
}
```

500	Internal Server Error

```
{
    "message": String
}
```

8.1.4. Módulo de ejecuciones

Tabla 8.12. Definición API crear ejecución

POST **/models/:modelId/executions**

Parameter

modelId ID del modelo

Body application/json

```
{
    "input_data": Object
}
```

Response application/json

201 Created

```
{
    "message": String,
    "id": String,
    "prediction": Number
}
```

403 Forbidden

```
{
    "message": String
}
```

404 Not Found

```
{
    "message": String
}
```

500 Internal Server Error

```
{
    "message": String
}
```

Tabla 8.13. Definición API listar ejecuciones

GET **/models/:modelId/executions**

Parameter

modelId	ID del modelo

Response application/json

200 OK

```
[
    {
        "id": String,
        "model_id": String,
        "username": String,
        "created_at": String,
        "input_data": Object,
        "prediction": Number,
        "real_value": Number,
        "returned_credits": Number
    }
]
```

500 Internal Server Error

```
{
    "message": String
}
```

Tabla 8.14. Definición API ver ejecución

GET	/models/:modelId/executions/:executionId
Parameter	
modelId	ID del modelo
executionId	ID de la ejecución
Response	application/json
200	OK

```
{
    "id": String,
    "model_id": String,
    "username": String,
    "created_at": String,
    "input_data": Object,
    "prediction": Number,
    "real_value": Number,
    "returned_credits": Number
}
```

404 Not Found

```
{
    "message": String
}
```

500 Internal Server Error

```
{
    "message": String
}
```

Tabla 8.15. Definición API añadir valor real a una ejecución

PUT	/models/:modelId/executions/:executionId
Parameter	
modelId	ID del modelo
executionId	ID de la ejecución
Body	application/json

```
{
    "real_value": Number
}
```

Response	application/json
200	OK

```
{
    "message": String
}
```

404	Not Found

```
{
    "message": String
}
```

500	Internal Server Error

```
{
    "message": String
}
```

8.1.5. Módulo de estadísticas

Tabla 8.16. Definición API obtener estadísticas

GET **/stats**

Response application/json

200 OK

```
{
    "general_stats": {
        "total_executions": Number,
        "total_credits_consumed": Number,
        "mean_credits_per_execution": Number,
        "std_credits_per_execution": Number,
        "total_added_credits": Number,
        "mean_added_credits": Number,
        "std_added_credits": Number,
        "latest_execution": Object,
        "earliest_execution": Object,
        "median_credits_per_execution": Number,
        "max_credits_per_execution": Number,
        "min_credits_per_execution": Number
    },
    "user_stats": {
        "total_credits": Number,
        "total_executions": Number,
        "mean_credits_per_execution": Number,
        "std_credits_per_execution": Number,
        "total_credits_consumed": Number,
        "total_added_credits": Number,
        "mean_added_credits": Number,
        "std_added_credits": Number,
        "median_credits_per_execution": Number,
        "max_credits_per_execution": Number,
        "min_credits_per_execution": Number,
        "most_used_model": Object
    }
}
```

404 Not Found

```
{
    "message": String
}
```

500 Internal Server Error

```
{
    "message": String
}
```

8.2 Pruebas adicionales del servicio

8.2.1. Módulo de usuarios

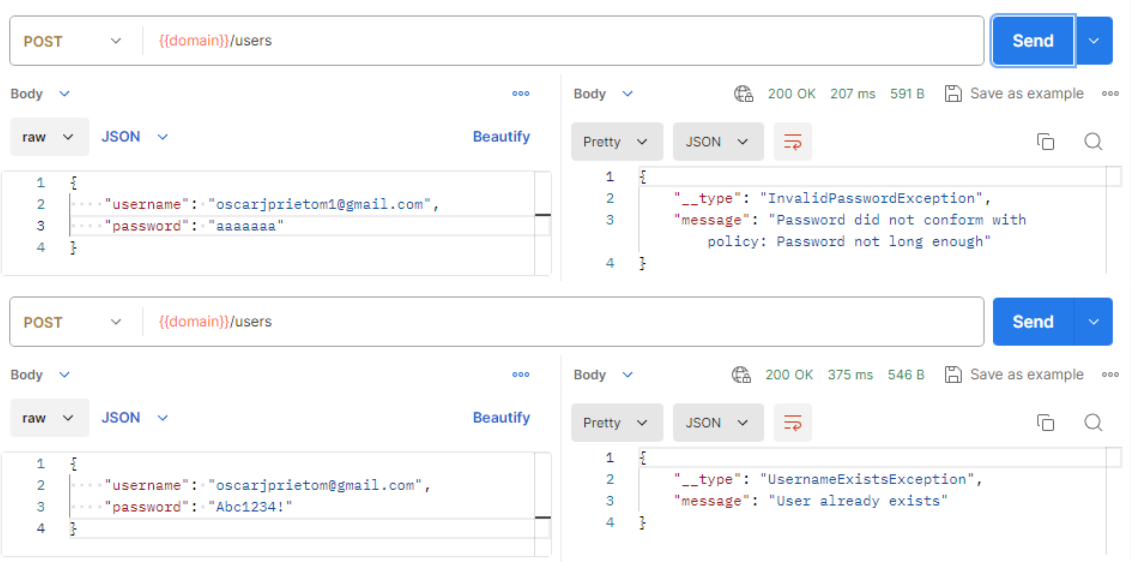

Figura 8.1. Pruebas adicionales crear usuario

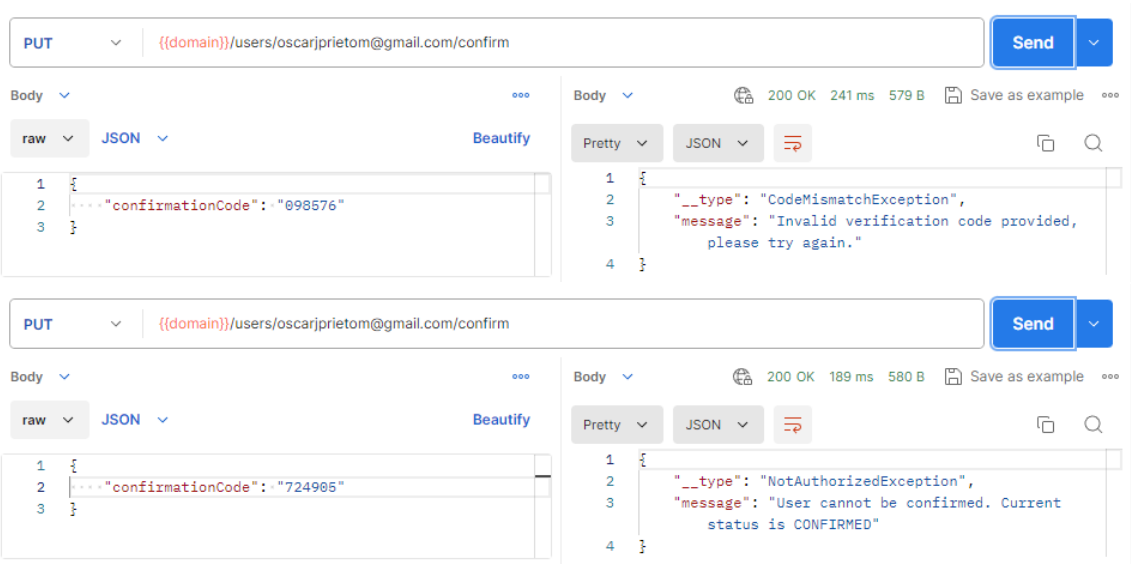

Figura 8.2. Pruebas adicionales confirmar usuario

POST {{domain}}/users/oscarjprieto@gmail.com/forgotPassword Send

Body none

This request does not have a body

Body 200 OK 190 ms 566 B Save as example

Pretty JSON

```
1 {
2     "__type": "UserNotFoundException",
3     "message": "Username/client id combination not
          found."
4 }
```

PUT {{domain}}/users/oscarjprietom@gmail.com/forgotPassword/confirm Send

Body raw JSON Beautify

```
1 {
2     "confirmationCode": "AAAA",
3     "password": "aaaaaa"
4 }
```

Body 200 OK 188 ms 579 B Save as example

Pretty JSON

```
1 {
2     "__type": "CodeMismatchException",
3     "message": "Invalid verification code provided,
          please try again."
4 }
```

PUT {{domain}}/users/oscarjprietom@gmail.com/forgotPassword/confirm Send

Body raw JSON Beautify

```
1 {
2     "confirmationCode": "223039",
3     "password": "aaaaaa"
4 }
```

Body 200 OK 281 ms 590 B Save as example

Pretty JSON

```
1 {
2     "__type": "InvalidPasswordException",
3     "message": "Password does not conform to policy:
          Password not long enough"
4 }
```

Figura 8.3. Pruebas adicionales restablecer contraseña

POST {{domain}}/users/oscarjprietom@gmail.com/login Send

Body raw JSON Beautify

```
1 {
2     "password": "aaaaaaa"
3 }
```

Body 200 OK 407 ms 557 B Save as example

Pretty JSON

```
1 {
2     "__type": "NotAuthorizedException",
3     "message": "Incorrect username or password."
4 }
```

Figura 8.4. Pruebas adicionales iniciar sesión

8.2.2. Módulo de créditos

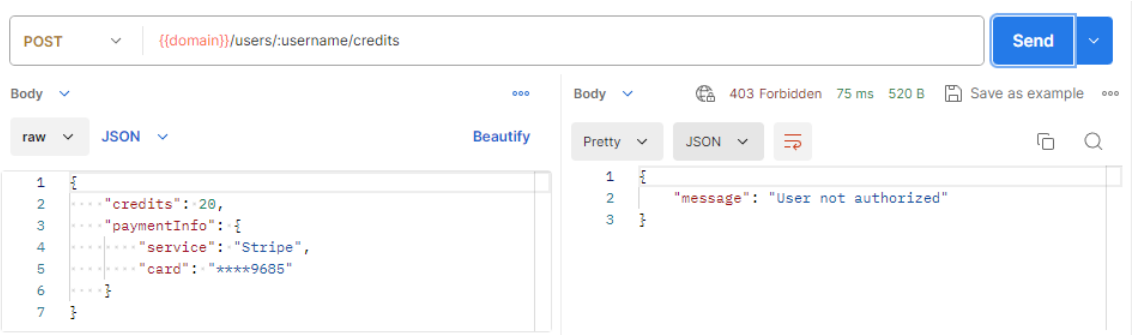

Figura 8.5. Pruebas adicionales agregar créditos

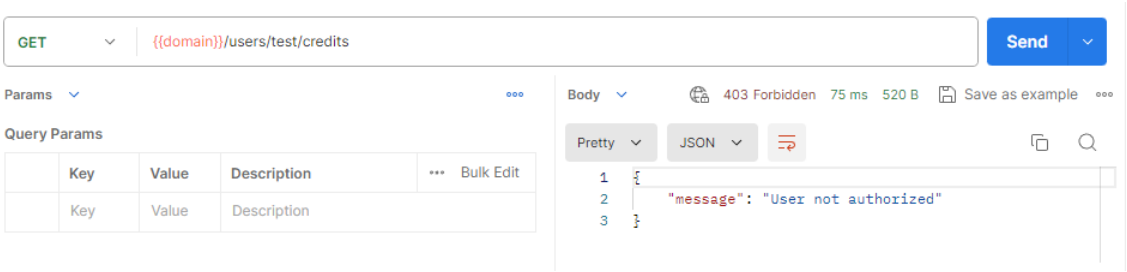

Figura 8.6. Pruebas adicionales obtener créditos

8.2.3. Módulo de modelos

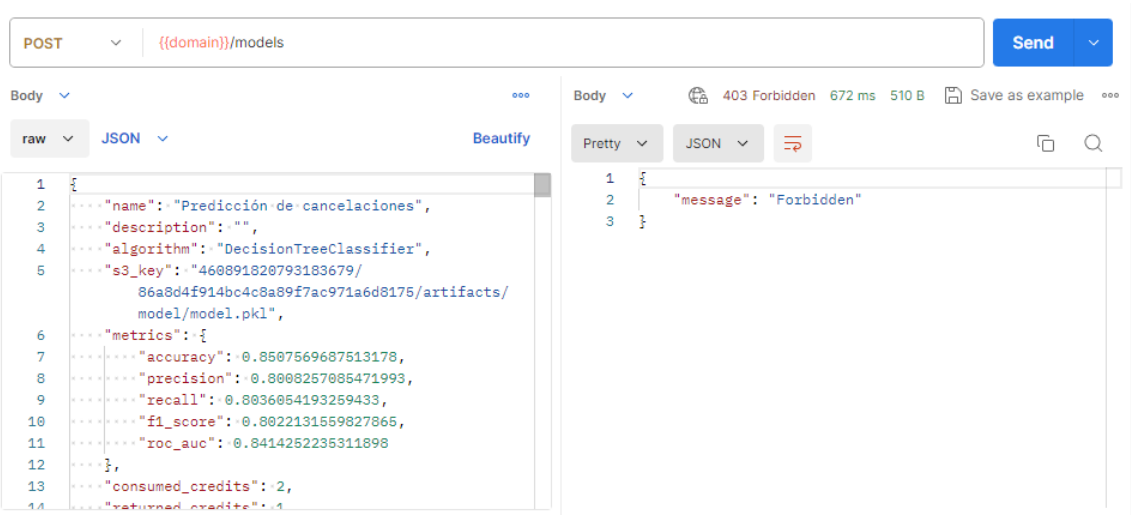

Figura 8.7. Pruebas adicionales crear modelo

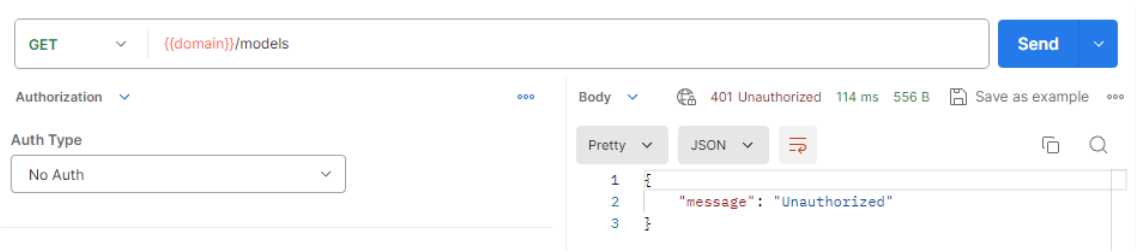

Figura 8.8. Pruebas adicionales listar modelos

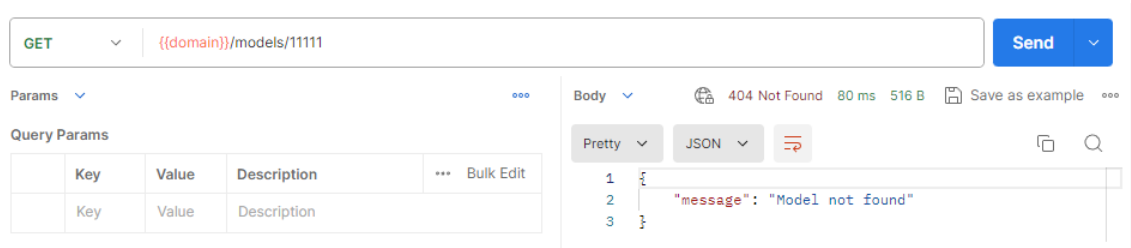

Figura 8.9. Pruebas adicionales obtener modelo

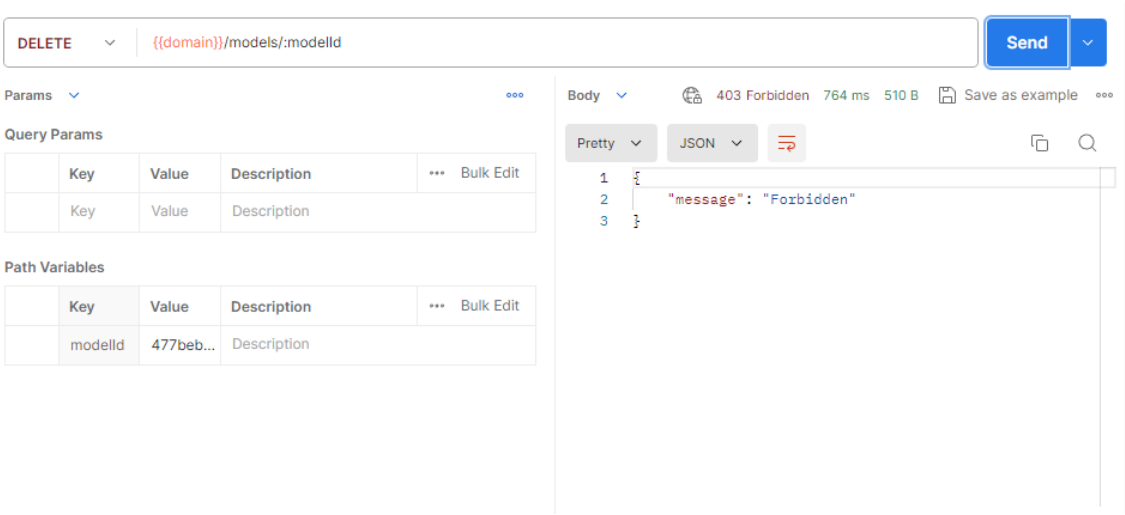

Figura 8.10. Pruebas adicionales eliminar modelo

8.2.4. Módulo de ejecuciones

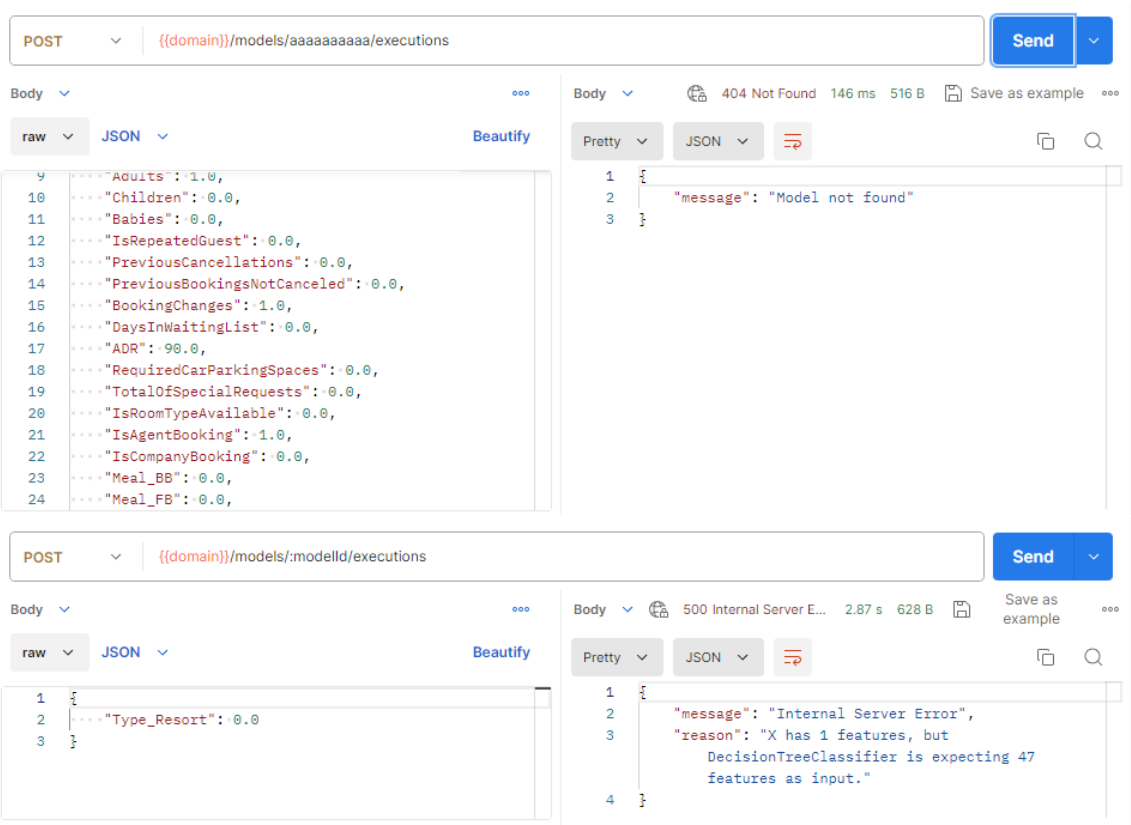

Figura 8.11. Pruebas adicionales crear ejecución

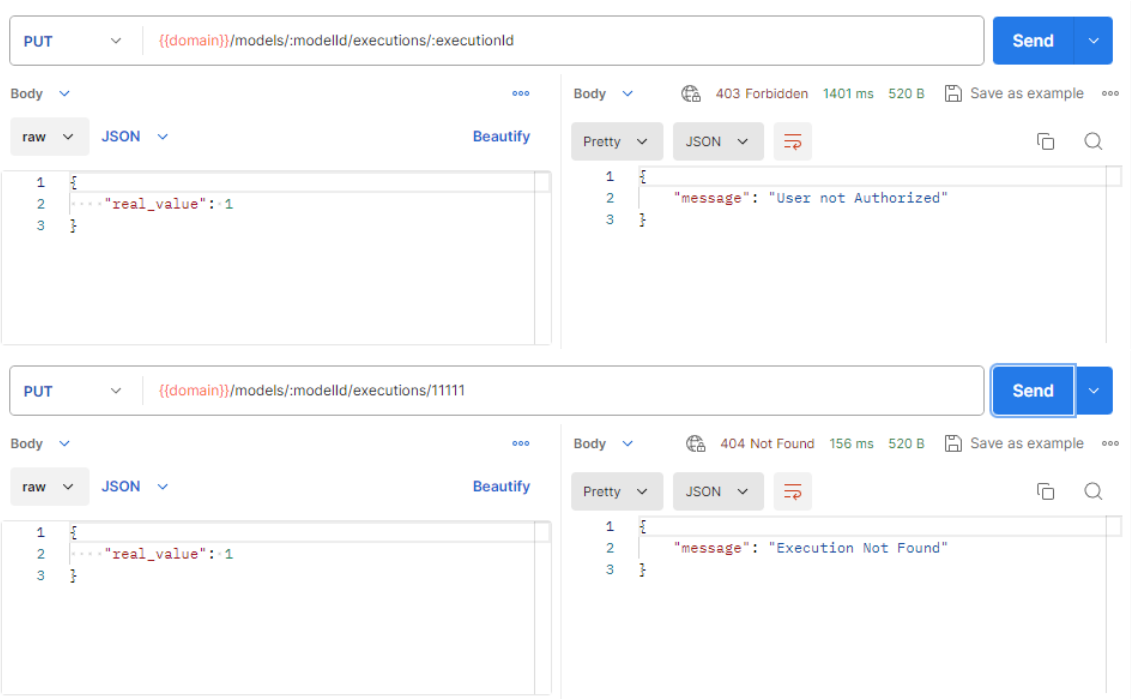

Figura 8.12. Pruebas adicionales añadir valor verdadero