Graduado en Ingeniería Informática

Universidad a Distancia de Madrid

Departamento de Ingeniería Informática

TRABAJO DE FIN DE GRADO

Desarrollo de una aplicación web para la predicción de la demanda de coches eléctricos en España.

Autor: Cintia García Garcés

Director: Antonio Martín Garré

MADRID, FEBRERO DE 2024

Declaración de originalidad:

El autor de este trabajo, Cintia García Garcés, con D.N.I. 71152250W, declara que el contenido de este trabajo de fin de grado es original, y en el caso de haber utilizado las ideas o contenidos de otros trabajos de otros autores están convenientemente citados, de acuerdo con las normas de citación establecidas.

El número de palabras de este trabajo, excluyendo los anexos es: 14.474

AGRADECIMIENTOS

Me gustaría expresar mi profundo agradecimiento, en primer lugar, a mi familia. A mi marido, Antonio y a mis hijas, Sofía y Helena, quienes siempre han estado dispuestos a brindarme su ayuda. Su apoyo constante, palabras de aliento y comprensión han sido fundamentales para superar los desafíos y mantenerme enfocada en mi objetivo.

También deseo expresar mi gratitud a mis padres y hermana, quienes siempre me han alentado a seguir adelante y perseguir mis metas. Su confianza en mí, sus sabios consejos y su apoyo constante me han ayudado durante todos estos años.

Además, me gustaría reconocer a mis compañeros de universidad, en particular a Alicia, Cristian, Luis y José Luis, quienes han estado a mi lado durante todo el recorrido. Su apoyo mutuo, compañerismo y colaboración han sido fundamentales para superar los desafíos académicos juntos.

No puedo olvidar mencionar a todos los profesores del grado de Informática de la UDIMA. Su amplio conocimiento y su valiosa orientación han sido fundamentales para mi crecimiento académico. Agradezco a todos aquellos que han sido mis profesores a lo largo de este camino, así como a aquellos que han tomado diferentes rumbos en sus carreras. Su dedicación y compromiso con la enseñanza han dejado una huella duradera en mi formación académica.

Por último debo agradecer a mi tutor, Antonio Maartin Garre, quien me ha acompañado durante todo el proceso de este trabajo de fin de grado. Su apoyo, orientación experta y disposición para ayudarme en todo momento han sido invaluables. Estoy profundamente agradecida por su dedicación y apoyo.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Resumen:

En este proyecto se ha llevado a cabo el desarrollo de una aplicación web que utiliza tecnologías avanzadas como algoritmos de aprendizaje automático e inteligencia artificial para predecir las ventas de vehículos eléctricos en un horizonte de 10 años.

Además, se ha implementado un chat que se ha conectado a un asistente creado específicamente para este propósito. El asistente ha sido entrenado para leer archivos de entrada necesarios para ser capaz de contestar a las preguntas que el usuario le haga sobre los datos históricos.

Este enfoque no solo ha permitido alcanzar el objetivo principal de mostrar las predicciones de ventas, sino que también ha dado lugar a la creación de una infraestructura flexible que puede ser adaptada para realizar predicciones en otros ámbitos y para diferentes empresas o negocios. De esta manera, se ha establecido una base sólida que puede ser reutilizada y ajustada según las necesidades específicas de cada caso.

Palabras clave:

Vehículos eléctricos, datos, predicción, aprendizaje automático, inteligencia artificial

Abstract:

In this project, the development of a web application has been carried out, which utilizes advanced technologies such as machine learning algorithms and artificial intelligence to predict the sales of electric vehicles over a 10-year horizon.

Additionally, a chat has been implemented, which is connected to a specifically created assistant. The assistant has been trained to read input files required to answer user questions regarding historical data.

This approach has not only enabled the achievement of the main objective of displaying sales predictions but has also resulted in the creation of a flexible infrastructure that can be adapted for making predictions in different domains and for various companies or businesses. Thus, a solid foundation has been established, which can be reused and adjusted according to the specific needs of each case.

Keywords:

Electric vehicles, data, prediction, machine learning, artificial intelligence

CAPÍTULO 1: INTRODUCCIÓN

Este documento redacta el trabajo de fin de grado de Ingeniería Informática de la mención en aplicaciones empresariales: la predicción de la demanda de coches eléctricos en España y la creación de cuadros de mando utilizando técnicas de aprendizaje automático. El diseño de cuadros de mando permite visualizar y analizar los resultados de manera efectiva.

Estas contribuciones son fundamentales para comprender y mejorar la adopción de la movilidad eléctrica, brindando una visión precisa y efectiva de los datos a través de representaciones visuales intuitivas.

1.1. Planteamiento del problema

En este capítulo se aborda el problema principal que motiva la realización de este trabajo fin de grado: la insuficiente infraestructura de carga de vehículos eléctricos en España, en respuesta a la creciente demanda de este tipo de vehículos.

La creciente relevancia de los vehículos eléctricos en la industria del automóvil está impulsada por varios factores. El factor más importante es el aumento de conciencia sobre las cuestiones medioambientales en nuestra sociedad y la necesidad de reducir las emisiones de gases. Los vehículos eléctricos ofrecen una alternativa más limpia y sostenible a los vehículos con motor de combustión, contribuyendo con un transporte más ecológico y reduciendo las emisiones de CO2.

A medida que aumenta la venta de coches eléctricos en España resulta crucial abordar los desafíos de infraestructura de carga asociados, especialmente cuando se considera la marcada diferencia entre las áreas urbanas y rurales. En este contexto, resulta evidente

una importante disparidad en términos de infraestructura de estaciones de carga de vehículos eléctricos, siendo las zonas rurales las más afectadas por esta desventaja.

Uno de los desafíos clave es la falta de infraestructura en muchas regiones de España. La limitación de estaciones de carga no sólo dificulta la adopción generalizada de vehículos eléctricos, sino que también plantea una barrera significativa para los consumidores interesados en su compra y uso diario. En las áreas urbanas, generalmente se encuentran más estaciones de carga disponibles, lo que brinda una mayor comodidad y accesibilidad para los usuarios de este tipo de automóviles. Por el contrario, en las zonas rurales, la infraestructura de carga es notablemente escasa, lo que limita la capacidad de los residentes para poder acceder y utilizar vehículos eléctricos de manera efectiva.

Para superar este desafío, es necesario poner énfasis en el desarrollo y expansión de las redes de infraestructura de carga. Esto incluye aumentar el número de estaciones de carga en espacios públicos, zonas residenciales, lugares de trabajo y en las carreteras.

1.2. Objetivo general y específicos

1.2.1 Objetivo general

El objetivo final de esta aplicación es proporcionar modelos predictivos que ayuden a las partes interesadas a anticipar las tendencias del mercado, optimizar la asignación de recursos y tomar decisiones estratégicas para satisfacer la creciente demanda de coches eléctricos en las diferentes regiones de España.

El propósito principal de este proyecto es desarrollar una infraestructura en la nube que procese y analice datos en tiempo real que permita visualizar, a través de una interfaz, las predicciones de la demanda de coches eléctricos mostrados en cuadros de mandos.

1.2.2 Objetivos específicos

Los objetivos específicos que se plantean son los siguientes:

Desarrollar un sistema de recolección de datos relevantes para la predicción de la venta de coches eléctricos. Esto implica identificar las fuentes de información pertinentes, como datos de ventas históricas, tendencias del mercado, factores socioeconómicos y políticas gubernamentales relacionadas con la movilidad sostenible.

Implementar algoritmos y modelos de predicción basados en técnicas de inteligencia artificial y aprendizaje automático. Estos modelos utilizan los datos recolectados para generar pronósticos precisos de la demanda futura de coches eléctricos. Se explorarán diferentes enfoques como regresión, series de tiempo y modelos predictivos basados en redes neuronales.

Diseñar una interfaz intuitiva y accesible para el usuario. La interfaz permitirá a las partes interesadas interactuar con los resultados de las predicciones y acceder a los informes y gráficos generados. Además, se implementarán funcionalidades adicionales como la posibilidad de ajustar parámetros de entrada y simular escenarios hipotéticos.

Evaluar y validar la precisión de los modelos de predicción mediante pruebas y comparaciones con datos reales. Se analizará el rendimiento de la aplicación en términos de exactitud de las predicciones y capacidad de adaptación a cambios en el entorno y condiciones del mercado.

Proporcionar la documentación detallada sobre el desarrollo y funcionamiento de la aplicación web. Pudiendo incluir manuales de usuario y descripción de los algoritmos implementados, explicación de la metodología utilizada y recomendaciones para su uso práctico.

Asimismo, se identificarán las dificultades y desafíos asociadas a esta predicción, tales como la falta de datos confiables y la complejidad de los factores influyentes.

1.3 Justificación

La falta de infraestructura adecuada para la carga de vehículos eléctricos presenta un obstáculo significativo para la adopción de estos vehículos en las carreteras españolas. Es fundamental conocer la demanda futura y planificar de manera adecuada la infraestructura de estaciones de carga para asegurar una transición exitosa hacia una movilidad eléctrica y sostenible.

Este trabajo ayudará a generar conocimiento y respaldar la toma de decisiones de las partes interesadas. Asimismo, se espera que los resultados obtenidos puedan ser utilizados en investigaciones futuras y contribuyan al avance de la movilidad eléctrica.

1.4. Alcance

Se creará una aplicación web que permita a los usuarios visualizar la predicción de la demanda de coches eléctricos a través de una interfaz intuitiva y fácil de utilizar. Esta aplicación se desarrollará con el objetivo de ofrecer a las partes interesadas, una herramienta que les permita prever las tendencias del mercado y tomar decisiones estratégicas basándose en la información proporcionada.

La aplicación web proporcionará modelos predictivos basados en datos históricos y variables relevantes, que permitirá estimar la demanda futura de coches eléctricos. Los modelos ayudarán a anticipar tendencias y patrones del mercado, lo que resulta valioso para la planificación de recursos, toma de decisiones y la optimización de la infraestructura.

Además, como parte del proyecto, se llevará a cabo un análisis de las estaciones de carga existentes en España. Esto incluirá la evaluación de la disponibilidad, ubicación de las estaciones de carga, así como la identificación de posibles limitaciones. Se propondrán recomendaciones y estrategias para mejorar la infraestructura de carga.
Se realizará la recopilación y análisis de datos relevantes para respaldar los resultados y contribuir a la transición de la movilidad eléctrica.

Capítulo 2: Estado de la Cuestión

2.1. Revisión de la bibliografía

En este apartado, se llevará a cabo un análisis de la bibliografía existente relacionada con la predicción de la demanda de coches eléctricos.

Con el auge de la movilidad eléctrica, numerosos estudios y publicaciones científicas han abordado este tema, proporcionando una visión más acertada de la situación en España.

Por ejemplo, acorde a la publicación de Diana Arrastia en el año 2021 en un artículo de la Vanguardia [2], redacta los factores que influyen en la intención de compra por parte de los españoles. Basándose en el estudio realizado por coches.net [10], los conductores no se plantean la transición a un vehículo eléctrico por las siguientes limitaciones: dificultad de encontrar estaciones de recarga (62%), baja autonomía (59%), precio (54%). Queda reflejado que habría que mejorar la infraestructura de carga en España para que los españoles decidan dar el paso a la movilidad eléctrica.

Según Victoria Fuentes en su artículo publicado el 13 de agosto de 2023 en motorpasion.com [4], destaca que España posee 20.243 puntos de recarga de acceso público. El estudio identifica las áreas con mayor y menor densidad de puntos de carga, permitiendo comprender las limitaciones y las oportunidades existentes en el desarrollo de esta infraestructura. Sin embargo, España se queda en la cola del ranking en cuanto a infraestructura de recarga de acceso público en los países de la Unión Europea.

2.2. Conceptos medulares

En esta sección se presentan los conceptos medulares asociados con la predicción de la demanda de coches eléctricos.

La demanda de coches eléctricos se refiere a la cantidad de vehículos eléctricos que los consumidores están dispuestos a adquirir en un cierto período de tiempo.

Los modelos predictivos son herramientas que ayudan a estimar la demanda futura, se basan en datos históricos y variables relacionadas.

2.3. Estudio de trabajos, publicaciones relacionadas

En este apartado, se lleva a cabo un análisis exhaustivo de los trabajos y publicaciones existentes relacionados con el desarrollo de una aplicación web que prediga la demanda futura de coches eléctricos. Se justifica la relevancia de este proyecto al considerar tanto la creciente importancia de los vehículos eléctricos como los objetivos específicos que este trabajo fin de carrera busca alcanzar. Además, es importante tener en cuenta que en Europa se está produciendo un cambio significativo en el sector automotriz, ya que se dejará de fabricar coches diésel y de gasolina en el año 2035 y se establecerán prohibiciones para su circulación en carreteras en un futuro cercano. Estos cambios regulatorios refuerzan aún más la necesidad de desarrollar una aplicación web que prediga la demanda futura de coches eléctricos y resalta la relevancia y oportunidad del presente proyecto en el contexto actual.

En el artículo “Un informe pronostica que en 2030 habrá casi 27 millones de vehículos eléctricos” publicado por ESMARTCITY en julio de 2019 [11], se proporciona información relevante sobre el crecimiento del mercado de vehículos eléctricos. Este

informe basado en un estudio de la consultora “MarketsandMarkets”, destaca que se espera un crecimiento exponencial en el número de vehículos eléctricos en todo el mundo.

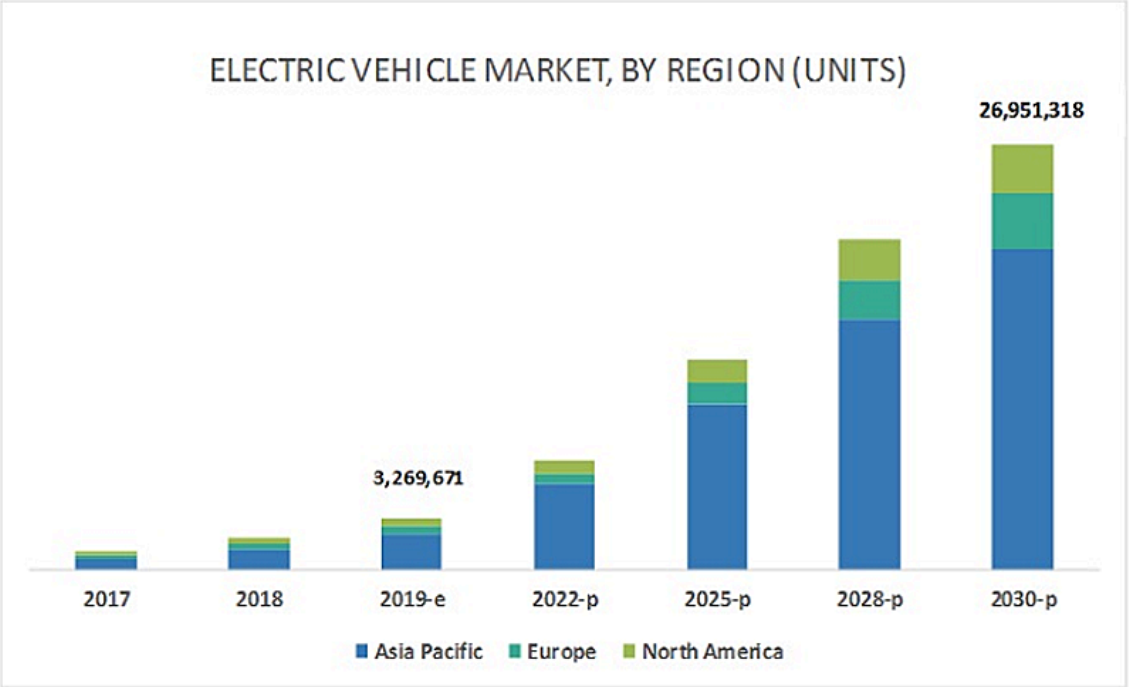


Figura 1: Gráfico de crecimiento del mercado de vehículos eléctricos por región hasta 2030

Se espera que el mercado europeo experimente un crecimiento constante debido al desarrollo de las infraestructuras necesarias para vehículos eléctricos.

Aunque la gráfica muestra el crecimiento de los vehículos eléctricos hasta el año 2030, se podría mejorar implementando un cuadro de mandos basado en aprendizaje automático. Permitiendo obtener pronósticos más precisos y en tiempo real sobre el crecimiento del mercado.

El cuadro de mandos permitiría a los usuarios visualizar de una manera interactiva y comprensible los resultados de las predicciones. Pudiendo explorar diferentes escenarios y ajustar las variables para obtener una visión más acertada de la evolución del mercado de vehículos eléctricos.

El estudio desarrollado por la organización sin ánimo de lucro de Ecodes "Infraestructura de recarga para vehículos eléctricos en España" [7] , proporciona un análisis detallado, evaluación y recomendaciones para impulsar la movilidad eléctrica y garantizar con éxito la transición a la movilidad eléctrica en España. En el informe se destaca la necesidad de una infraestructura de recarga adecuada y la distribución equitativa en todo el país. En el resumen se apunta que España se encuentra rezagada en comparación con otros países europeos en términos de penetración de vehículos eléctricos, principalmente debido a la falta de estaciones de carga y su desigual distribución en el mapa español.

	Año 2022	Año 2023	Variación 22-23	% variación 22-23
Puntos de carga	4.041	5.527	1.486	36,77 %
Conectores 0-49 kW	10.179	13.952	3.773	37,07 %
Conectores 50-149 kW	1.494	2.190	696	46,59 %
Conectores baja potencia (<150 kW)	11.673	16.142	4.469	38,28 %
Conectores 150-249 kW	114	323	209	183,33 %
Conectores > 249 kW	48	181	133	277,08 %
Conectores alta potencia (>149 kW)	162	504	342	211,11 %
Total conectores	11.835	16.646	4.811	40,60 %

Tabla 1: Puntos de carga en áreas urbanas.

Fuente: Ecodes

El informe resalta la necesidad de aumentar la cantidad de cargadores de alta potencia y fomentar las inversiones públicas y privadas para continuar desarrollando la red de cargadores en todo el territorio español.

El estudio de Ecodes proporciona un análisis exhaustivo de la infraestructura de recarga en España, incluyendo datos detallados de las estaciones de carga por Comunidad Autónoma y provincia, así como los diferentes tipos de conectores disponibles. Además, se presenta de manera visual un mapa que muestra los ejes entre los puntos de carga por Comunidad Autónoma. La información contenida en el estudio es muy valiosa, sin embargo, lamentablemente, está presentada en un formato pdf estático que limita la capacidad del usuario de interactuar con los datos.

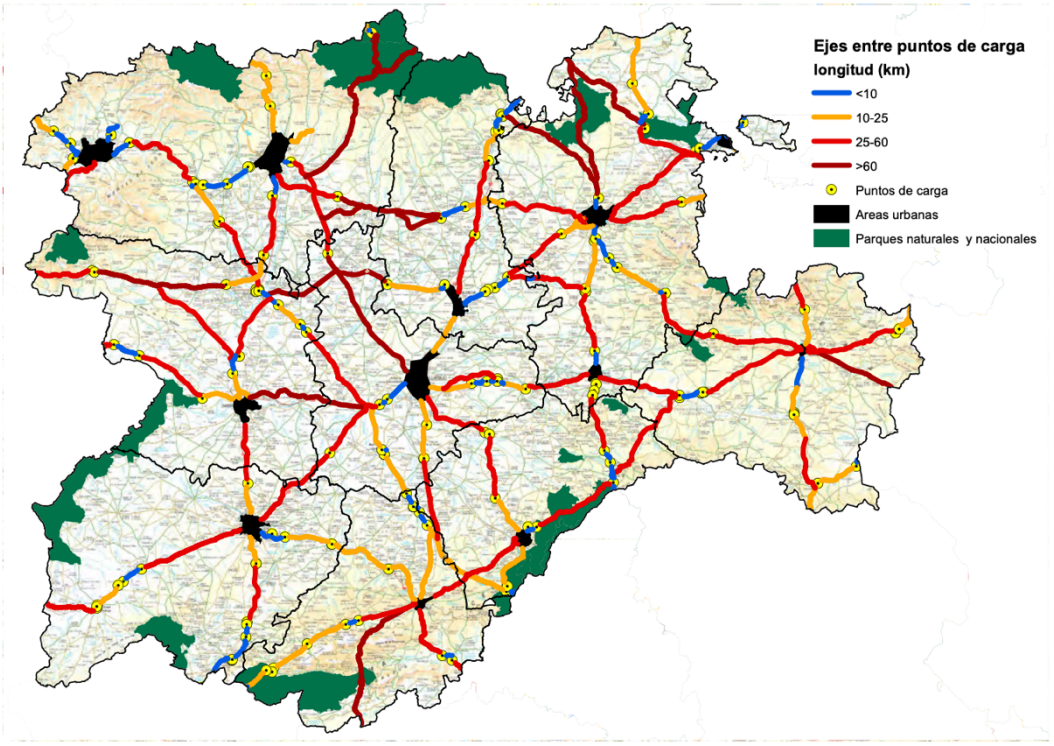


Figura 2: Eje entre puntos de carga en Castilla y León

Fuente: Ecodes

Para mejorar la experiencia del usuario y brindar una forma más interactiva de acceder a esta información, se propone desarrollar una aplicación web como parte del trabajo de fin de grado. Esta aplicación web permitirá a los usuarios explorar la infraestructura de carga de manera más activa y personalizada.

Algunas de las características que se podrían incluir en este trabajo fin de grado:

1. Búsqueda y filtrado: Permitir a los usuarios buscar estaciones de carga por ubicación geográfica, tipo de conector, potencia.
2. Visualización interactiva: En lugar de un mapa estático, la aplicación web podría incluir un mapa interactivo donde los usuarios pueden seleccionar las regiones y obtener información más detallada.
3. Integración de datos en tiempo real: Si es posible, se podría considerar la opción de incluir datos en tiempo real. Esto podría actualizar la información sobre nuevas estaciones de recarga, cambios en la disponibilidad o cualquier otra actualización relevante que mejore la precisión y utilidad de la aplicación.

De esta forma se proporcionará una experiencia más enriquecedora y permitiría a los usuarios aprovechar al máximo la información sobre las estaciones de recarga en el territorio español.

Capítulo 3: Metodología, Planificación y costes

3.1. Metodología

En el área del desarrollo de productos de software, es de vital importancia emplear las metodologías adecuadas para permitir una efectiva planificación, una gestión eficiente de los recursos y la entrega de un producto de calidad. Las metodologías adecuadas para la implementación de la aplicación web serán: Kanban y Scrum.

La combinación de ambas metodologías ayuda a obtener un desarrollo eficiente del producto de software, desde el MVP (Producto Mínimo Viable), hasta las iteraciones cortas para la implementación de nuevas funcionalidades.

3.1.1. Metodología Kanban

La metodología Kanban consiste en la gestión visual del flujo de trabajo. Permite a los equipos de trabajo tener una visión clara de las tareas que se encuentran pendientes, las que están en progreso y las que ya están finalizadas. Facilita así la visualización de los cuellos de botella y optimización del flujo de trabajo impactando de esta manera en una mayor eficiencia y productividad. En este proyecto se utilizará Kanban para llevar a cabo el desarrollo del producto de software hasta alcanzar un MVP (Producto Mínimo Viable).

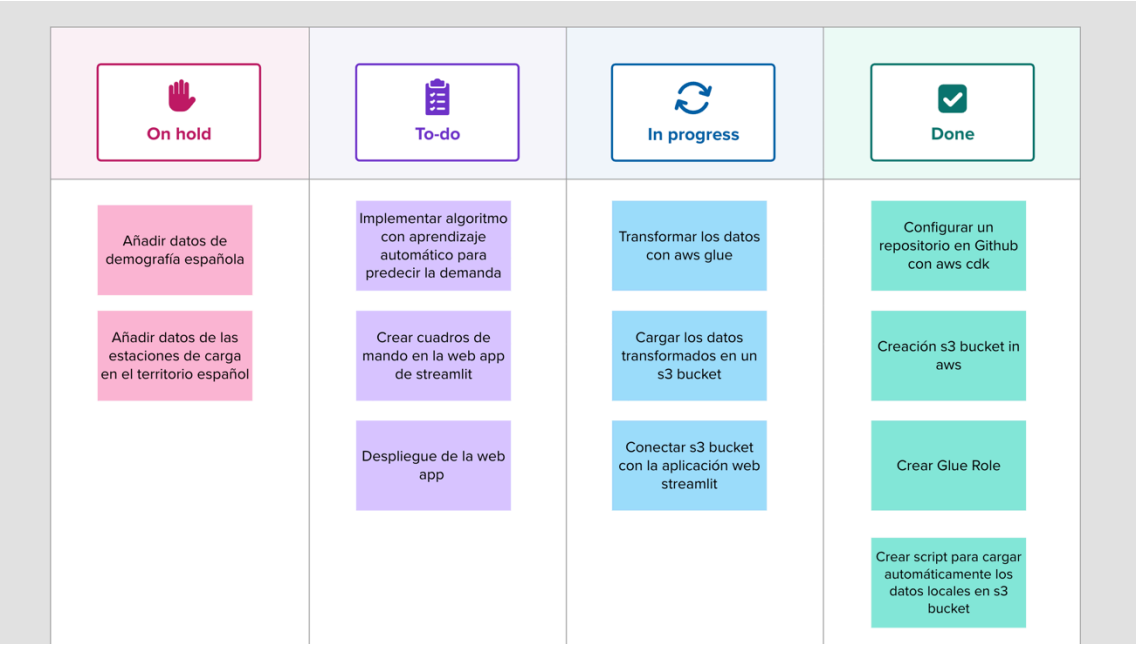


Figura 3: Flujo de trabajo en el tablero Kanban

3.1.2. Metodología Scrum

La metodología Scrum es un enfoque ágil centrado en la entrega iterativa e incremental del software. El trabajo está dividido en 'sprints', son períodos de tiempo fijos y cortos, usualmente de una a dos semanas. En cada sprint, se seleccionan y completan las funcionalidades más urgentes. Cuando termina el sprint, se realiza la revisión del trabajo entregado y se recolecta la retroalimentación de clientes y usuarios. Esta metodología promueve la mejora continua del desarrollo.

3.2. Planificación

En el siguiente epígrafe, se establecen los objetivos y se desarrollarán las etapas necesarias para la implementación del cuadro de mando. Los pasos clave del proceso de planificación son los siguientes:

3.2.1 Definición de los objetivos

La primera etapa es definir los objetivos del cuadro de mando. Se puede visualizar la demanda de coches eléctricos en diferentes regiones geográficas, la identificación de patrones de consumo o la evaluación de factores externos que puedan afectar a la demanda.

3.2.2 Identificación de los requisitos

Se recopilan los requisitos necesarios para la creación del cuadro de mando. Se identifican las métricas clave, los datos necesarios para la predicción de la demanda y los elementos visuales que son requeridos para presentar la información de manera efectiva. También habrá que tener en cuenta los requisitos de seguridad y privacidad de los datos.

3.2.3 Diseño de la arquitectura

Se define la arquitectura necesaria para la implementación de la aplicación web. Se determinan las fuentes de datos necesarias, como pueden ser los datos de ventas, datos demográficos, infraestructura de carga de vehículos eléctricos. Se diseñarán los algoritmos y los modelos predictivos que se aplicarán para la creación del cuadro de mando. Así como las tecnologías necesarias en la implementación.

3.2.4 Desarrollo del cuadro de mando

Esta etapa consiste en la implementación del cuadro de mando. Se lleva a cabo la extracción, la transformación y carga de datos para prepararlos para el análisis. Los modelos predictivos desarrollados se integran en el cuadro de mando. Además, se implementarán paneles de visualización que mostrarán las métricas y resultados de la predicción.

3.2.5 Pruebas y validación

Una vez terminada la anterior etapa, se realizan las pruebas exhaustivas para asegurar que la aplicación web funciona como se espera. Se verificará la precisión de las predicciones realizadas y se compararon con los datos históricos disponibles verificando su eficacia. Asimismo, se realizan pruebas de rendimiento para garantizar que el cuadro de mando es capaz de manejar gran cantidad de volúmenes de datos y tener un nivel de respuesta adecuado.

3.2.6 Implementación y despliegue

Una vez que se ha completado la anterior etapa, se produce su implementación en el entorno de producción. Se realiza la configuración necesaria para que la aplicación web sea accesible y adecuada para los usuarios finales.

3.2.7. Mantenimiento y mejora continua

Después de la implementación y despliegue, se establece un plan de mantenimiento y monitorización para asegurar el correcto funcionamiento de la aplicación. Se trabajará en futuras actualizaciones y mejoras según sea necesario, y se recuperará retroalimentación de los usuarios para su posterior estudio. La recolección de la

información proveniente de los usuarios servirá para identificar oportunidades de mejora.

3.2.8 Planificación inicial

Las etapas de la planificación se dividirán en cinco fases principales, como se muestra en la siguiente tabla, donde se aprecia la cantidad de horas que se deberán invertir en cada una de ellas. A continuación se detallan las etapas y las horas correspondientes:

Etapa	Subetapa	Horas
Definición de objetivos		35
Identificación de Requisitos		25
Arquitectura		50
Desarrollo	Implementación	130
	Pruebas	15
	Mantenimiento	10
Documentación		58
Total		323

Tabla 2: Etapas de planificación

3.3. Costes

Los costes asociados con el proyecto de desarrollo de una aplicación web de predicción de la demanda de coches eléctricos, se pueden dividir en tres categorías: costes de planificación, diseño y desarrollo, costes de implementación y despliegue y por último los costes de mantenimiento.

3.3.1 Costes de planificación, diseño y desarrollo:

3.3.1.1 Recursos Humanos

Son los costes relacionados con el personal involucrado en el desarrollo de la aplicación. En mi caso estaría formado por un desarrollador junior, trabajando como Ingeniero de datos, analista de datos y programador fronted.

3.3.1.2 Licencias y Software

Estos costes están asociados con la adquisición de licencias de software necesarias para el desarrollo y funcionamiento de la aplicación, así como cualquier otro software de terceros que sea necesario.

3.3.1.3 Datos y Fuentes Externas:

Costes relacionados con la obtención y adquisición de los datos relevantes para el desarrollo del proyecto, como los datos de ventas de automóviles eléctricos, datos demográficos, etc.

3.3.2 Costes de implementación y despliegue:

Costes asociados con la infraestructura necesaria para el desarrollo y la implementación de la aplicación como pueden ser los servidores, alojamiento web, bases de datos, servicios en la nube.

3.3.3 *Costes de mantenimiento:*

Son los costes asociados con las posibles actualizaciones de la aplicación web, así como su correcto funcionamiento.

3.3.3.1 Recursos Humanos:

Estimación mensual de los costes relacionados con el soporte y el mantenimiento de la aplicación.

3.3.3.2 Infraestructura tecnológica

Referido al mantenimiento continuo de la aplicación. Costes de servidores y alojamiento web, costes de aws (s3, Athena).

Elemento de Coste	Coste/Hora	Horas	Coste Total Estimado
Planificación, diseño y desarrollo			
- Desarrollador	25	130	1.500
- Licencias y Software	20	70	1.400
OpenAI	0.002	100.000	200
- Datos y Fuentes Externas	15	10	300
Implementación y despliegue			
- Servicio y alojamiento web			2.400
- Coste de aws (s3, Athena)			Variable
Mantenimiento			
- Recursos Humanos	25	16	400
- Servicio y alojamiento web			2.400
- Coste de aws (s3, Athena)			Variable
Total sin I.V.A.			8.600
I.V.A.		21%	1.806
Total con I.V.A.			10.406

Tabla 3: Costes

CAPÍTULO 4: DESARROLLO

En este apartado, se aborda el desarrollo de la aplicación web. El desarrollo de esta aplicación web abarca las siguientes etapas: requisitos, análisis, diseño, implementación, despliegue, rendimiento, uso y mantenimiento. Seguidamente, se presenta una visión de cada una de las etapas.

4.1. Requisitos

4.1.1. Educción u obtención de requisitos

Para la obtención de requisitos se llevaron a cabo diferentes métodos de recolección de información. Se realizó una revisión de la documentación existente sobre movilidad eléctrica, revisión de estadísticas de tráfico y vehículos registrados, así como un análisis de tendencias y avances tecnológicos.

Se llevó a cabo una revisión exhaustiva de la documentación existente sobre la movilidad eléctrica, informes de mercado y estudios relacionados con la demanda de vehículos eléctricos en España. Especialmente la búsqueda se centró en aquellos informes que abordan la demanda actual y las proyecciones futuras. La investigación permitió recopilar información valiosa sobre las necesidades y expectativas de los usuarios, así como las peculiaridades que se deberían tener en cuenta al desarrollar la aplicación.

Se realizó un análisis sobre las tendencias y los avances tecnológicos existentes en el área de la movilidad eléctrica, identificando de esta manera requisitos emergentes y oportunidades de mejora en la aplicación. Se exploraron blogs especializados y

publicaciones científicas, para obtener conocimientos actualizados sobre el mercado y las expectativas de los usuarios.

Por último, se obtuvieron datos de organismos gubernamentales o entidades relevantes sobre el registro de vehículos eléctricos en España. Se tuvo en cuenta patrones de crecimiento, áreas geográficas con mayor adopción y otros indicadores que puedan influir en la demanda de vehículos eléctricos.

La combinación de ambas técnicas permitió obtener los requisitos necesarios para el desarrollo de la web. Estos requisitos sirvieron de base para satisfacer las principales funcionalidades del sistema, teniendo en cuenta las necesidades de los usuarios.

4.1.2. Análisis de requisitos

Una vez realizada la anterior fase se procedió al análisis de los requisitos obtenidos. Estos requisitos se clasificaron como: funcionales, no funcionales, usabilidad y de mantenimiento. El análisis de requisitos proporcionó una base sólida para realizar el diseño y el desarrollo de la aplicación web.

4.1.3. Representación y documentación de requisitos

4.1.3.1. Requisitos Funcionales (RF):

- Requisito 1.1 (RF_1.1): Recopilación de datos:
  El sistema debe tener capacidad para poder acceder y recolectar datos en tiempo real sobre las ventas de coches eléctricos en las diferentes regiones de España.
- Requisito 1.2 (RF_1.2): Procesamiento de datos históricos

La aplicación web deberá contar con un algoritmo para procesamiento de datos eficiente que pueda manejar grandes cantidades de datos históricos de ventas y que prediga la demanda futura de coches eléctricos en España.

- Requisito 1.3 (RF_1.3): Visualización de los datos
  El usuario deberá ser capaz de visualizar los resultados de las predicciones a través de gráficos interactivos y mapas que muestran la distribución geográfica de la demanda.

4.1.3.2. Requisitos No Funcionales (RNF):

- Requisito 2.1 (RNF_2.1): Rendimiento
  La aplicación web deberá ser capaz de procesar y analizar grandes cantidades de conjuntos de datos en un tiempo razonable, pudiendo garantizar una respuesta rápida a los usuarios.
- Requisito 2.1 (RNF_2.2): Seguridad
  Para proteger la integridad de los datos se deben implementar medidas de seguridad robustas.

4.1.3.3. Requisitos de Usabilidad (RU):

- Requisito 3.1 (RU_3.1): Interfaz intuitiva
  Con respecto a la interfaz, debe ser intuitiva y fácil de usar. Esto permitirá a los usuarios navegar de manera sencilla por la aplicación y comprender los resultados de las predicciones.
- Requisito 3.2 (RU_3.2): Compatibilidad con dispositivos
  Para garantizar una buena experiencia de usuario, la aplicación web deberá ser compatible con una amplia gama de dispositivos. Esto implica que la aplicación debe ser capaz de funcionar sin problemas en diferentes tipos de dispositivos,

cómo teléfonos móviles, tabletas y computadoras de escritorio, independientemente del sistema operativo con el que funcionen.

- Requisito 3.3 (RU_3.3): Compatibilidad con navegadores
  La aplicación web desarrollada debe ser compatible con navegadores populares como Google Chrome, garantizando así que los usuarios puedan acceder sin ningún problema y utilizar la aplicación de manera efectiva sin importar qué navegador están utilizando para visualizar los datos.

4.1.3.4. Requisitos de Mantenimiento (RM):

- Requisito 4.1 (RM_4.1): Actualización de datos
  Sería conveniente implementar un mecanismo automático que se encargue de actualizar regularmente los datos históricos y de ventas de coches eléctricos.
- Requisito 4.2 (RM_4.2): Escalabilidad
  La arquitectura de la aplicación deberá ser escalable para poder manejar un aumento en el número de usuarios y la cantidad de datos en el tiempo.

4.2. Análisis

4.2.1 Análisis de requisitos

En este epígrafe, se trata de analizar los requisitos del apartado anterior para definir cómo van a interactuar con los distintos componentes de la aplicación y cómo se implementarán.

4.2.1.1 RF_1.1 – Recopilación de datos

Es necesario una capacidad de acceso y recolección de datos en tiempo real sobre las ventas de coches eléctricos en las diferentes regiones de España. Para ello se necesita un

sistema que acceda a las fuentes de datos y se encargue de almacenar la información en el s3 bucket en AWS destinado para dicha recolección. Para el posterior procesamiento de datos la información deberá estar clasificada y organizada.

Es importante señalar que actualmente no es posible realizar esta recolección de datos en tiempo real debido a la falta de una Api pública proporcionada por la Dirección General de Tráfico (DGT) que permita extraer los datos mediante una solicitud GET a un api. Por lo tanto, la extracción de los datos se llevará a cabo de forma manual mediante la descarga de los archivos pertinentes, los cuales posteriormente serán cargados en el bucket de Amazon s3. Ver extracción de los datos desde la página de la DGT en Anexo A2.

4.2.1.2 RF_1.2 – Procesamiento de datos históricos

Para procesar los datos históricos se puede considerar el uso de algoritmos de aprendizaje automático, en particular los clasificados como modelos de regresión. Una vez que se hayan preparado los datos se pueden considerar diferentes algoritmos de regresión. Entre ellos se encuentra el algoritmo de regresión lineal, que es ampliamente utilizado para establecer relaciones lineales entre variables. Para el análisis de series temporales, se puede considerar el algoritmo Prophet. Prophet es un algoritmo de código abierto desarrollado por Facebook que permite desarrollar pronósticos precisos y flexibles en series temporales.

En cuanto a los algoritmos de redes neuronales, si se dispone del tiempo y los recursos adecuados, pueden ser considerados para tareas de regresión más complejas. Las redes neuronales tienen la capacidad de aprender patrones no lineales en los datos, por lo que les convierte en una opción poderosa para modelar relaciones más sofisticadas. Sin embargo, hay que tener en cuenta que la elección del uso de redes neuronales puede demandar una mayor capacidad de procesamiento y un conjunto de datos más extenso para su entrenamiento adecuado.

4.2.1.3 RF_1.3 – Visualización de los datos

Se propone utilizar cuadros de mando para cumplir las expectativas de este requisito. Esta herramienta proporciona una interfaz intuitiva para generar visualizaciones y permite a los usuarios interactuar con los datos.
Estos gráficos pueden incluir, gráficos de líneas, gráficos de barras, gráficos de dispersión, dependiendo de los datos y el tipo de análisis que se quiera mostrar. Los usuarios podrán explorar los datos y ajustar los parámetros de visualización según sus necesidades.
Además, se integrarán mapas interactivos que se pueden utilizar para mostrar la distribución por Comunidades Autónomas.

4.2.1.4 RF_2.1 – Rendimiento

Este requisito se centra en garantizar un rendimiento óptimo de la aplicación para procesar y analizar los conjuntos de datos de manera eficiente, pudiendo proporcionar una rápida respuesta a los usuarios. Se podrá medir el rendimiento mediante la aplicación de métricas, como pueden ser, medición del tiempo de respuesta y la escalabilidad.

4.2.1.5 RF_2.2 – Seguridad

Se está trabajando con un conjunto de datos públicos proporcionados por la Dirección General de Tráfico de España. Dado que estos datos son públicos y no contienen información personal identificable, no es necesario cumplir con el RGPD (Reglamento General de Protección de Datos) ni aplicar medidas de seguridad tan rigurosas como en el caso de datos personales, según lo establecido en el Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo. [8] Por este motivo, la encriptación puede no ser

estrictamente necesaria. Sin embargo, si se requiere una capa adicional de seguridad, se puede considerar el uso de encriptación para proteger la integridad y la confidencialidad de los datos durante su almacenamiento y transmisión.

Aunque los requisitos de seguridad no son tan rigurosos como en el caso de datos personales, aún es importante implementar medidas de seguridad adecuadas para proteger la integridad y disponibilidad de los datos públicos, y garantizar su uso ético y responsable.

4.2.1.6 RF_3.1 – Interfaz intuitiva

Se necesita crear una estructura clara y organizada que facilite la navegación y comprensión de los resultados de las predicciones. Se utilizarán elementos como menús desplegables, gráficos, botones para mejorar la experiencia de usuarios.
La estructura de menú proporciona a los usuarios una visión clara donde poder acceder a las diferentes funcionalidades de la aplicación.

4.2.1.7 RF_3.2 – Compatibilidad con dispositivos

El fronted se debe adaptar automáticamente al tamaño de la ventana del navegador, lo que permite la compatibilidad con diferentes tamaños de pantalla y dispositivos. Sin embargo, será imprescindible realizar pruebas en diferentes dispositivos y tamaños de pantalla para verificar que la interfaz se ajuste y se vea bien en cada uno de ellos. Esto asegurará que los usuarios puedan acceder y utilizar la aplicación de manera efectiva, independientemente del dispositivo que utilicen.

Por lo tanto, hay que verificar que sea compatible con una variedad de navegadores más populares como Google Chrome, Mozilla Firefox, Safari y Microsoft.

4.2.1.8 RF_3.2 – Compatibilidad con navegadores

Por lo tanto, hay que verificar que sea compatible con una variedad de navegadores adicionales. Esto implica realizar pruebas exhaustivas en diferentes navegadores populares, como Chrome, Firefox, Safari, y otros, para asegurarse de que la funcionalidad y apariencia del sistema se mantengan consistentes en todos ellos. Es fundamental garantizar una experiencia de usuario óptima y sin problemas, independientemente del navegador utilizado por el usuario.

4.2.1.9 RF_4.1 – Actualización de datos

Es importante evaluar la importancia y la frecuencia de actualización de los datos. Habría que investigar si existe una necesidad real por parte de los usuarios de mantener los datos actualizados de manera periódica.
Al no existir una API disponible para obtener y descargar los datos automáticamente, es necesario analizar otras fuentes para poder mantener la información actualizada. Actualmente, para poder procesar los datos se requiere la descarga manual de los datos directamente desde la fuente, DGT. [12]

4.2.1.10 RF_4.2 – Escalabilidad

Es importante comprender el alcance del crecimiento esperado para poder determinar cómo la arquitectura de la aplicación puede satisfacer estas necesidades.
La aplicación se divide en componentes separados y escalables permitiendo escalar cada componente de manera independiente según sea necesario. La escalabilidad se aborda mediante el escalado horizontal de la aplicación Streamlit y el manejo eficiente del almacenamiento de datos en S3 buckets en AWS.

Realizar pruebas de carga y monitorización del rendimiento del sistema asegura que se cumplan los requisitos de escalabilidad establecidos.

4.2.2 Análisis de usuarios

Es fundamental realizar un análisis exhaustivo de los usuarios potenciales que van a utilizar la aplicación con el principal fin de comprender plenamente sus necesidades y expectativas. Al tener en cuenta los requerimientos necesarios de los usuarios se podrá implementar una aplicación que proporcione una experiencia de usuario satisfactoria.

4.2.2.1 Compradores de coches eléctricos

Estos usuarios representan a las personas físicas que están interesados en adquirir un coche eléctrico para uso personal. Las necesidades que necesitan ser cubiertas pueden ser las siguientes:

- Acceso a datos y análisis de las tendencias del mercado de vehículos eléctricos en las diferentes regiones de España.
- Visualización de proyecciones de crecimiento y evolución de la demanda de coches eléctricos a lo largo de los años.

4.2.2.2 Fabricante de coches eléctricos

Estos usuarios van a representar a las empresas encargadas en la fabricación de coches eléctricos. Sus necesidades pueden incluir:

- Acceso a análisis y a los datos sobre la evolución de la demanda de vehículos eléctricos en España, incluyendo las estimaciones de ventas, segmentación del mercado y las preferencias del consumidor.

- Proyecciones de crecimiento de la demanda a largo plazo para poder tomar decisiones estratégicas sobre la producción, inversión y expansión de la capacidad de fabricación.

4.2.2.3 Empresas de alquiler de vehículos

Las empresas de alquiler de vehículos son usuarios dedicados al alquiler de una flota de vehículos y pueden estar interesadas en la adquisición de coches eléctricos. Sus necesidades relacionadas con la demanda de coches eléctricos podrían incluir:

- Análisis de mercado para identificar las oportunidades de negocio y la viabilidad de incorporar coches eléctricos a su flota de alquiler considerando las proyecciones futuras.
- Acceso a los datos sobre la demanda de coches eléctricos en diferentes regiones de España, incluyendo la información sobre la disponibilidad de las instalaciones de carga para poder soportar la operación de una flota de coches eléctricos.
- La predicción de la demanda permite que se tomen decisiones estratégicas sobre la incorporación de este tipo de vehículo a su flota, así como ofrecer a los clientes opciones más sostenibles con el medio ambiente.

4.2.2.4 Proveedores de servicios de carga

Estos usuarios son representados por las empresas que se encargan de proporcionar servicios de carga para vehículos eléctricos. Es importante que se puedan anticipar y poder satisfacer la demanda creciente de carga en el mercado. Sus necesidades pueden incluir:

- Analizar la evolución de la infraestructura de carga de vehículos eléctricos en España, incluyendo la ubicación y la distribución de las estaciones de carga existentes.
- Estudios de mercado para identificar las regiones de España con mayor demanda de potencial de servicios de carga, considerando la cantidad de vehículos eléctricos existentes por región, la densidad de población y patrones de movilidad.
- Proyecciones del crecimiento de la demanda de coches eléctricos a corto y largo plazo, para planificar la expansión de las instalaciones de servicio de carga y asegurar que los usuarios de servicios de carga puedan acceder a estos servicios sin problema.

4.2.2.5 Gobierno y entidades locales

Desempeñan un papel importante en el marco regulatorio y normativo referente a los servicios de carga para vehículos eléctricos. La participación del gobierno y las entidades locales es esencial para establecer leyes y reglamentos que promuevan la accesibilidad a la infraestructura de carga, así como garantizar la conformidad con las normativas medioambientales y demás regulaciones aplicables. Algunas necesidades podrían incluir:

- Los gobiernos pueden requerir el acceso a datos específicos proporcionados por la aplicación web para realizar el análisis y tomar decisiones informadas. Pudiendo incluir datos relacionados con la infraestructura de carga, la demanda de servicios de carga, la movilidad eléctrica, entre otros.
- Los gobiernos y entidades locales podrán utilizar la web como una herramienta para la planificación estratégica de la infraestructura de carga. Implicando que podrán analizar los datos proporcionados en los cuadros de mando e identificar áreas con necesidades insatisfechas, determinar la ubicación óptima para nuevas

estaciones de carga, evaluar la demanda futura y tomar decisiones sobre inversiones y políticas públicas relacionadas con la movilidad eléctrica.

- La aplicación web puede ser utilizada por los gobiernos como un medio de comunicación y divulgación de información que sea relevante para los ciudadanos y otras partes interesadas. Pudiendo incluir la publicación de mapas interactivos con la ubicación de las estaciones de carga, promoción de programas de incentivos para vehículos eléctricos, y la difusión de noticias y eventos relacionados con la movilidad eléctrica.

4.3. Diseño

En un mundo tan cambiante y más consciente de la importancia de cuidar el medio ambiente, donde la tendencia es una evolución a la movilidad sostenible, la demanda de coches eléctricos está experimentando un crecimiento significativo. Para comprender y anticiparse a esta demanda en el futuro, se ha desarrollado una aplicación web de predicción de demanda de coches eléctricos.

En el siguiente diagrama se ilustra una descripción general de alto nivel de los componentes principales de la canalización de datos y el enfoque para desarrollar la arquitectura de la canalización de alto nivel.

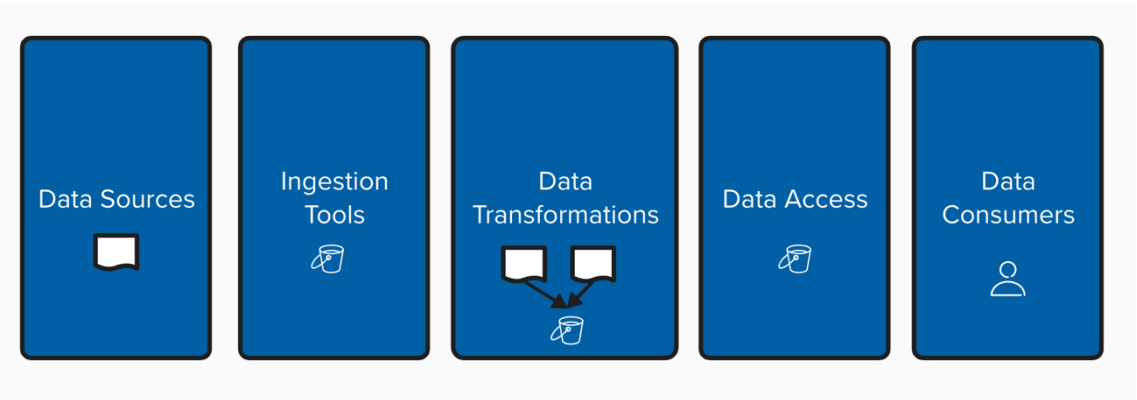


Figura 4: Arquitectura de alto nivel

La aplicación va a utilizar tecnologías avanzadas de AWS (Amazon Web Services) para almacenar y procesar grandes volúmenes de datos. Los datos brutos son descargados del portal estadístico de la DGT (Dirección General de Tráfico) y se almacenan de forma segura en un bucket de Amazon S3, denominado ‘raw-data’.

Mediante el uso de AWS Glue, los datos se transforman y se generan conjuntos de datos limpios que se van a almacenar en otro bucket de S3, denominado ‘clean-data’. Estos datos procesados y listos para su análisis se convierten en la base para nuestras predicciones de demanda.

Al utilizar AWS Glue para procesar los datos y cargarlos en ‘clean-data’ bucket es necesario primero configurar los permisos necesarios para que AWS Glue pueda acceder al bucket de Amazon S3 y realizar las tareas de extracción, transformación y carga (ETL).

Los permisos necesarios que se necesitan configurar son los siguientes:

1. Acceso a los buckets de S3:
   El servicio AWS Glue necesita tener permisos para leer los datos del bucket de S3, ‘raw-data’ que contiene los datos brutos en formato CSV. Se configura la política de acceso (IAM policy) para AWS Glue, garantizando permisos de lectura sobre el S3 bucket correspondiente.
2. Permisos de AWS Glue en el bucket destino:
   Para escribir los datos procesados en el bucket S3, ‘clean-data’, AWS Glue necesita permisos de escritura en dicho bucket
3. Permisos IAM para AWS Glue:
   Además de tener permisos de escritura y lectura en los buckets S3, AWS Glue necesita tener permisos de IAM para realizar tareas específicas, como crear y ejecutar trabajos de ETL, acceder a catálogos de datos.

Para ello se tiene que asignar un rol de IAM que tenga los permisos necesarios para realizar dichas operaciones.

Hay que tener en cuenta que para seguir buenas prácticas de seguridad se debe otorgar los permisos necesarios mínimos a AWS Glue. Esto garantiza la seguridad de los datos y evita la exposición innecesaria de recursos.

Los datos al estar almacenados en buckets en S3, ofrece flexibilidad, escalabilidad y acceso rápido a los datos necesarios para generar las predicciones y visualizaciones en la aplicación web.

La aplicación web, creada con Streamlit, ofrece una interfaz intuitiva y fácil de usar que permite a los usuarios acceder a los datos procesados y visualizarlos en atractivos dashboards. A través de gráficos interactivos y herramientas de visualización de datos, los usuarios pueden explorar y comprender la demanda de coches eléctricos en diversas regiones y periodos de tiempo.

Los usuarios pueden sumergirse en los datos y descubrir patrones, tendencias y relaciones relevantes para la demanda de coches eléctricos. Pueden generar visualizaciones personalizadas, seleccionar las regiones específicas o comparar diferentes periodos de tiempo para obtener una visión completa.
Se ofrecen herramientas de filtrado y exploración que brindan la flexibilidad de adaptar la visualización a las necesidades individuales.

En el siguiente gráfico se muestra la arquitectura de una forma visual:

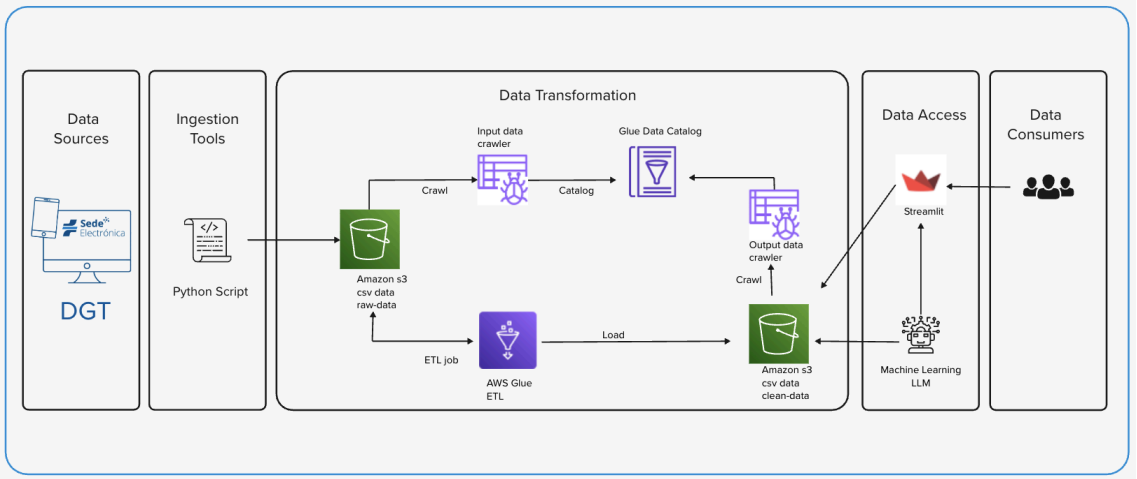


Figura 5: Arquitectura de software de alto nivel

4.4. Implementación

4.4.1 Descripción general

La implementación de la aplicación web se ha desarrollado utilizando AWS CDK (Cloud Development Kit) en Typescript. AWS CDK ha sido utilizado para crear y gestionar los recursos en la nube. [1]

La computación en la nube permite escalabilidad, eficiencia en costes, seguridad y automatización. Además, la computación en la nube permite a los usuarios proporcionar flexibilidad en el uso de recursos, acceso global a datos y aplicaciones, y la capacidad de adaptarse rápidamente a las necesidades cambiantes.

Se ha utilizado CDK para desarrollar el código necesario para crear los buckets en aws. Los S3 bucket donde se han almacenado los datos encriptados mediante el uso de una clave KMS (Key Management Service), se les ha llamado “raw-data-ve” y “clean-data-ve” respectivamente.

La carga automática al bucket de S3 y la transformación de datos han sido implementados utilizando Python. Como los ficheros no incluían ninguna fecha, se ha tenido que añadir lógica en el código para que los datos cargados en el S3 bucket se almacenen en diferentes carpetas según el año, y a su vez se crean nuevas carpetas para distinguir el mes. Además, se ha agregado la información de la fecha a los ficheros mediante la adición de dicha información en la última columna. Para lograrlo, se ha utilizado el siguiente código:

```
def encode_csv_to_unicode(file_path):

    # Leer el archivo CSV en un DataFrame de pandas
    df = pd.read_csv(file_path, encoding='latin-1')

    # Agregar nueva columna al archivo CSV
    df['Year'] = year
    df['month'] = str(month).zfill(2)

    # Guardar el DataFrame codificado en un nuevo archivo CSV
    encoded_file_path = 'encoded_file.csv'
    df.to_csv(encoded_file_path, index=False)

    return encoded_file_path

def upload_file_to_s3(bucket_name, file_path, s3_key, kms_key_arn):
    s3 = boto3.client('s3')

    # Subir el archivo al bucket de AWS S3
      s3.upload_file(file_path, bucket_name, s3_key, ExtraArgs={'SSEKMSKeyId':
kms_key_arn, 'ServerSideEncryption': 'aws:kms'})

    print(f"File uploaded to S3 bucket: s3://{bucket_name}/{s3_key}")

bucket_name = 'raw-data-ve-eu-central-1'
s3_key_prefix = 'data/'
kms_key_arn                                                              =
'arn:aws:kms:eu-central-1:766973746059:key/4ae649c9-0b3f-4080-8344-2b0c7696e44
a'
```

```
# Recorrer la estructura de directorios
for year in range(2016, 2024):
   for month in range(1, 13):
       # Generar la ruta del archivo basada en la estructura de directorios
                                                                file_path        =
f"/Users/I559673/Documents/Informatica/tfg_ve/data/{year}/{str(month).zfill(2)
}/InformePredefinido_Parque{year}{str(month).zfill(2)}.csv"

       # Comprobar si el archivo existe
       if os.path.exists(file_path):
           # Codificar el archivo CSV a Unicode
           encoded_file_path = encode_csv_to_unicode(file_path)

           # Generar la clave de S3 basada en la estructura de directorios
                                                                    s3_key      =
f"{s3_key_prefix}{year}/{str(month).zfill(2)}/informe_VE_{year}_{str(month).zf
ill(2)}.csv"

           # Subir el archivo codificado a AWS S3
                    upload_file_to_s3(bucket_name,  encoded_file_path,  s3_key,
kms_key_arn)
       else:
           print(f"File does not exist: {file_path}")
```

Figura 6: Función para cargar los datos al S3 bucket

Se ha utilizado el servicio de AWS Glue, un servicio de extracción, transformación y carga (ETL), para llevar a cabo la transformación de los datos. Se ha creado un rol en AWS Glue, que permite llevar a cabo la transformación de los datos almacenados en el S3 bucket. Mediante el desarrollo de un script en Python, se ha definido la lógica de transformación de los datos y se ha creado un flujo de trabajo automatizado en AWS Glue para ejecutar dicho script y cargar los datos transformados en el bucket de S3.

En el proceso de transformación de datos, se ha utilizado un Glue Crawler para descubrir la estructura y el esquema de los datos almacenados en el bucket S3. Esto permite que AWS Glue pueda inferir el esquema y facilita la manipulación de los datos durante el proceso de transformación.

Una vez que los datos han sido transformados con éxito utilizando AWS Glue, se han cargado nuevamente en el bucket S3, pero esta vez en una ubicación separada denominada ‘clean-data-ve’. Esta ubicación contiene los datos transformados y preparados para su uso en la aplicación web.

Para el desarrollo del fronted de la aplicación, se ha establecido una conexión segura con Streamlit. Streamlit es un framewok de Python diseñado para crear aplicaciones web interactivas de manera rápida y sencilla. Permite extraer los datos almacenados en el bucket de s3 y utilizarlos para generar visualizaciones, tablas u otros elementos interactivos en la interfaz de usuario de la aplicación. La conexión se ha establecido exitosamente siguiendo la guía proporcionada en la página de Streamlit. Esta guía explica cómo acceder de forma segura a archivos en AWS S3 desde la nube de la Comunidad de Streamlit. Utiliza la conexión de Streamlit, FilesConnection, la biblioteca s3fs y opcionalmente, la gestión de secretos de Streamlit. [5]

```python
def read_data_from_s3(bucket_name, conn):
        # Crear un objeto de sistema de archivos S3
        fs = s3fs.S3FileSystem()
        file_paths = fs.glob(f"{bucket_name}/*")


        # Leer los objetos y procesar los datos
        data_frames = []
        for file_path in file_paths:
            with fs.open(file_path, "rb") as file:
                # Read the object content, a csv file
                df = pd.read_csv(file)
                data_frames.append(df)

        # Concatenar todos los marcos de datos
        df = pd.concat(data_frames)

        return df
```

Figura 7: Función para leer los datos de un S3 bucket

La función mencionada anteriormente se encarga de leer los datos de múltiples archivos almacenados en un S3 bucket en AWS. Luego estos datos son visualizados y presentados a través de cuadros de mando.

Las tecnologías utilizadas en este proyecto, como AWS CDK, TypeScript, Python, SQL y Streamlit, se encuentran detalladamente explicadas en el anexo 1 de este documento, donde se proporciona información sobre su uso, características y cómo se han aplicado en el desarrollo de la solución.

4.4.2 Arquitectura de la aplicación

La aplicación web para predecir la demanda futura de coches eléctricos en España se ha diseñado siguiendo una arquitectura basada en componentes. Esta arquitectura permite una mayor modularidad, escalabilidad y mantenibilidad de la aplicación. La arquitectura se encuentra diseñada en la figura 5 de este trabajo. Posteriormente se procede a reflejar los componentes principales de la arquitectura de la aplicación.

4.4.2.1. Fronted

El fronted de la aplicación se ha desarrollado utilizando el framework Streamlit. Los usuarios interactúan con la interfaz de usuario a través de un navegador web. El fronted se comunica con los componentes del backend a través de la API de Streamlit que se conecta al bucket S3 para obtener y enviar datos de manera eficiente y segura. Esto permite una comunicación fluida entre fronted y backend, garantizando un intercambio de información sin problemas.

La flexibilidad en la adaptación a diferentes navegadores se debe en gran medida a los componentes y características proporcionadas por el framework Streamlit. Streamlit está diseñado para ser compatible con una amplia gama de navegadores, lo que asegura que la aplicación se pueda ejecutar sin problemas en los diferentes entornos de navegación.

Además, Streamlit ofrece una interfaz de usuario reactiva y responsiva, lo que significa que los elementos de la interfaz se ajustan automáticamente al tamaño de la pantalla del dispositivo en el que se esté utilizando la aplicación. Esto permite una visualización y navegación adecuadas tanto en pantallas grandes como en dispositivos móviles más pequeños.

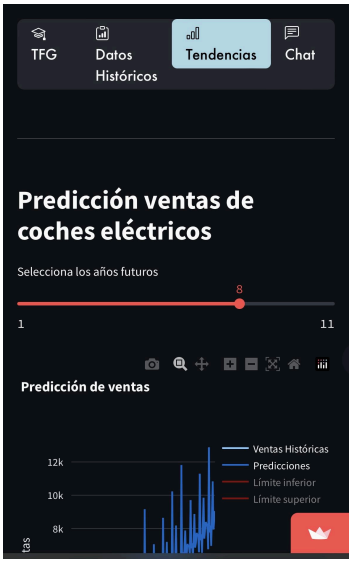


Figura 8: Aplicación web en pantalla de móvil

Adicionalmente, se ha incluido una versión en modo oscuro (dark mode) que ofrece la posibilidad a los usuarios de cambiar la apariencia de la interfaz basándose en sus preferencias.

Figura 9: Dark mode en la aplicación web

En cuanto a la navegación, se ha incorporado una barra lateral izquierda que contiene un menú con diversas opciones para que el usuario pueda seleccionar:

- Home: permite acceder a la página principal de la aplicación.
- Datos Históricos: proporciona acceso a los datos almacenados en el sistema, permitiendo a los usuarios explorar y analizar información histórica relacionada con la demanda de coches eléctricos.
- Tendencias: ofrece visualizaciones y análisis de tendencias actuales en el mercado de coches eléctricos, permitiendo a los usuarios obtener información relevante sobre el tema.
- Chat: brinda la posibilidad al usuario de interactuar de manera activa y realizar preguntas acerca de los datos históricos.

Para brindar una experiencia de usuario atractiva, se han utilizado colores cautivadores que generan un impacto visual positivo. Estos colores han sido seleccionados de una manera cuidada para crear una interfaz agradable y estimulante, lo que contribuye a una experiencia de usuario enriquecedora y atractiva.

4.4.2.2. Backend

Se ha utilizado un componente de almacenamiento basado en Amazon S3 para almacenar de forma segura los datos utilizados por los modelos de predicción. Este componente también se encarga de la carga y actualización de los datos a medida que se recopilan nuevos datos. Cuando se incorporan nuevos datos en Amazon S3, se sigue un procedimiento específico para mantener la organización y la estructura de los datos. En este caso, se procede a la creación de una nueva carpeta para cada año y mes correspondiente a los datos recopilados.

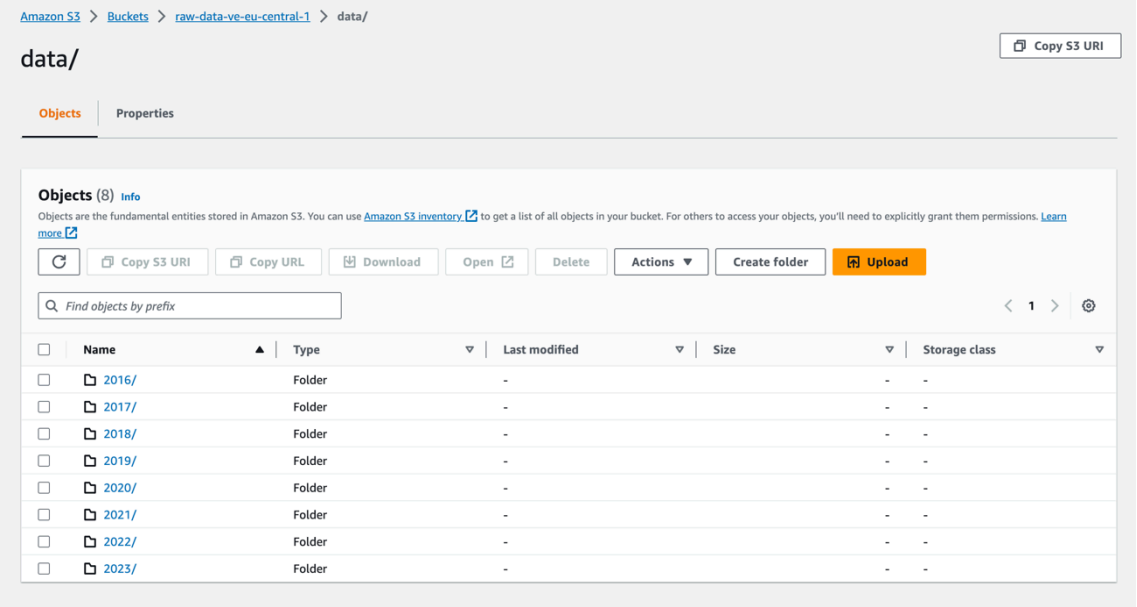


Figura 10: Almacenamiento de los datos en S3 bucket

La transformación de los datos se ha desarrollado mediante AWS Glue Studio, una herramienta visual y sin código que permite diseñar y ejecutar flujos de trabajo de ETL de manera intuitiva en AWS Glue.

Se establece la conexión a las fuentes de datos de origen, como en este caso S3. Se crean los flujos de trabajo en AWS Glue Studio y se incluyen las operaciones necesarias para la transformación de datos como puede ser filtrado, agregación, limpieza de datos. Se añade el destino donde van a ser cargados los datos ya transformados, en el proyecto se ha creado un nuevo s3 bucket denominado ‘clean-data-ve’

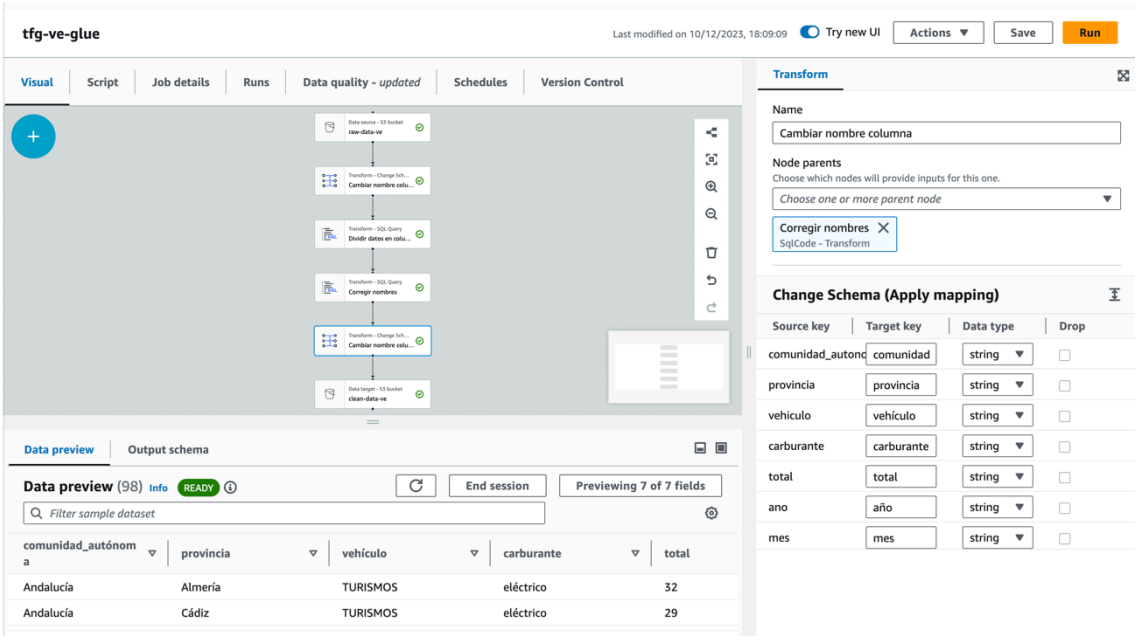


Figura 11: Transformación de los datos en AWS Glue

Los datos se han descargado manualmente desde el portal estadístico de la DGT [12] en formato CSV. Sin embargo, los datos obtenidos presentan algunos desafíos como se puede apreciar en la siguiente figura adjunta.

Es importante destacar que los datos descargados inicialmente desde la DGT no contienen las columnas relativas al año y al mes, las cuales resultan fundamentales para poder aplicar posteriormente los algoritmos necesarios de aprendizaje automático. Para poder solucionar esta limitación se ha tenido que añadir posteriormente dichas columnas mediante un script

La incorporación de las columnas “año” y “mes” es crucial para realizar un análisis temporal de los datos y aprovechar las funcionalidades de los algoritmos de aprendizaje automático que dependen de la información temporal.

El script utilizado ha permitido extraer la información de fecha existente en los datos originales y añadir dicha información en la última columna que representan el año y el mes correspondiente a cada registro. Esto asegura que los datos estén correctamente estructurados y preparados para aplicar los algoritmos que requieren información temporal.

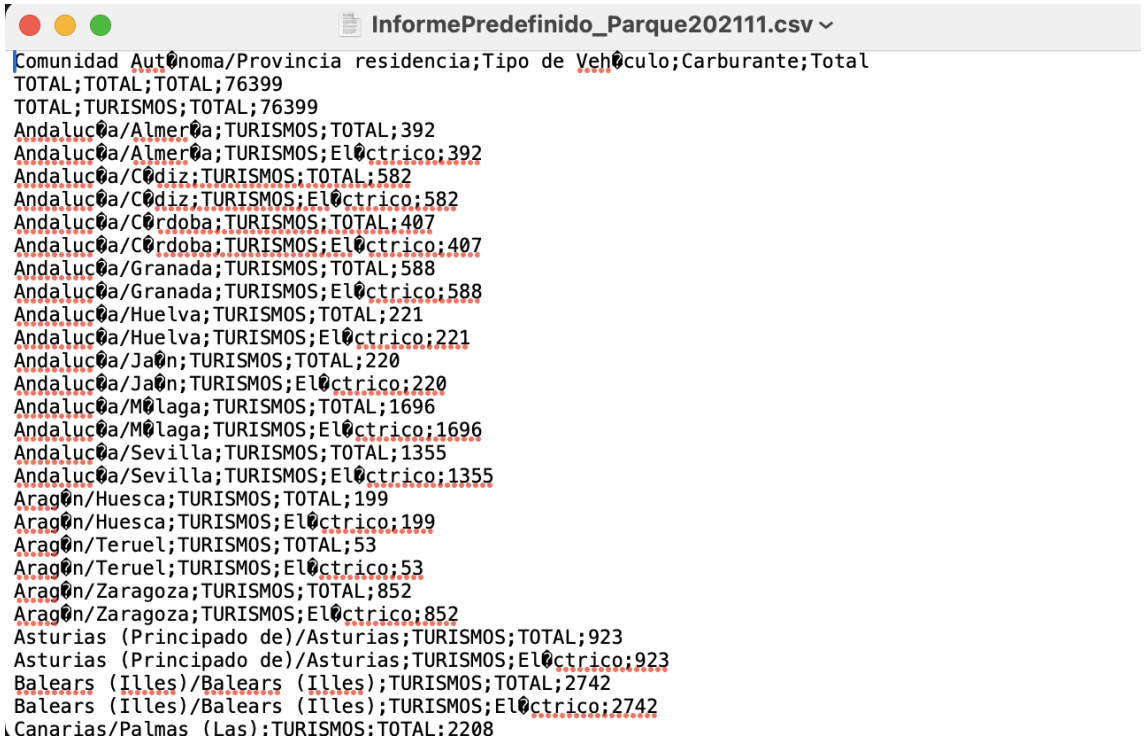


Figura 12: Datos obtenidos de la DGT en formato csv

Posteriormente, una vez que los datos han sido almacenados de forma segura en el componente de almacenamiento basado en Amazon S3 y siguiendo la estructura de carpetas por año y mes, se procede a realizar las tareas de transformación utilizando AWS Glue.

Las tareas de transformación con AWS Glue se llevan a cabo mediante la definición de trabajos (jobs) de ETL, donde se tienen que especificar las operaciones de transformación necesarias para convertir los datos con la estructura y el formato deseado.

En este caso, se han realizado las siguientes tareas con el fin de llevar a cabo la limpieza de los datos:

1. Extracción de los datos del Amazon S3, “raw-data-ve”: Para ello es necesario especificar la ubicación y los detalles del origen de datos, en este caso se especifica la ruta a la carpeta que contienen los datos que se quieren extraer. AWS Glue puede inferir el esquema de los datos automáticamente. En este caso es necesario tener creado un IAM role (Rol de Identidad y Acceso de AWS) con los permisos necesarios para acceder a los datos almacenados en S3, así como para realizar otro tipo de operaciones necesarias en el proceso de extracción, transformación y carga de datos.

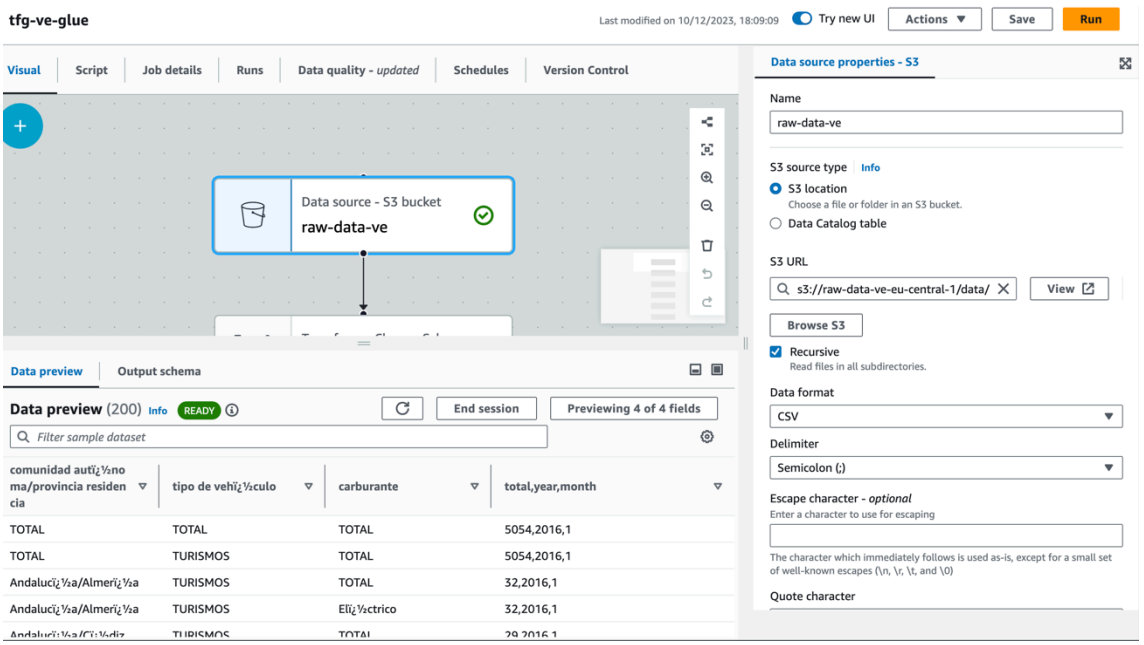


Figura 13: Datos extraídos del S3 bucket

2. Cambiar el esquema: Se procede al cambio del nombre de las columnas para posteriormente poder dividir la información en diferentes columnas.

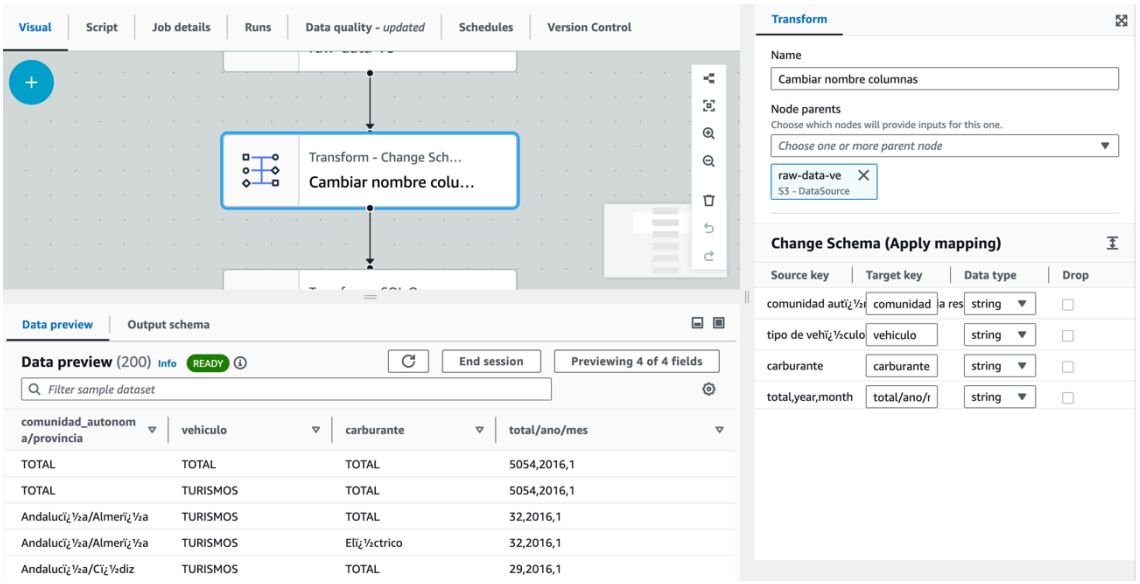


Figura 14: Transformación del nombre de las columnas

3. División de datos en columnas: Se realiza la división de los datos en columnas utilizando una query SQL. Se dividen los datos en columnas separadas utilizando la función Split y asigna los resultados a las nuevas columnas creadas. El resultado final mostrará el conjunto de datos con las columnas divididas y filtradas cuando la columna carburante no tenga un valor igual a ‘TOTAL’.

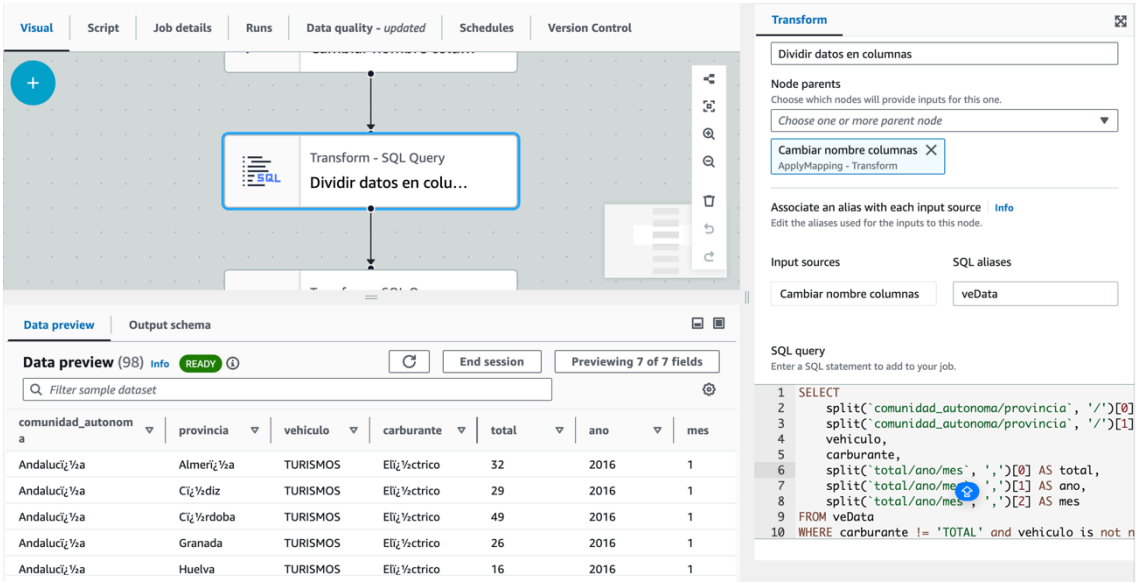


Figura 15: División de columnas y filtrado de datos

4. Transformación de datos: Para que los datos se ajusten al formato, la estructura o estándares requeridos antes de ser cargado en el destino final, se ha procedido a la corrección de estos. La transformación implica utilizar una lógica definida, como la cláusula CASE en la consulta de SQL, para modificar los valores de ciertas columnas y asignarles nuevos nombres corregidos. Esta etapa es crucial para garantizar la consistencia, calidad y coherencia de los datos antes de su carga en su destino final.

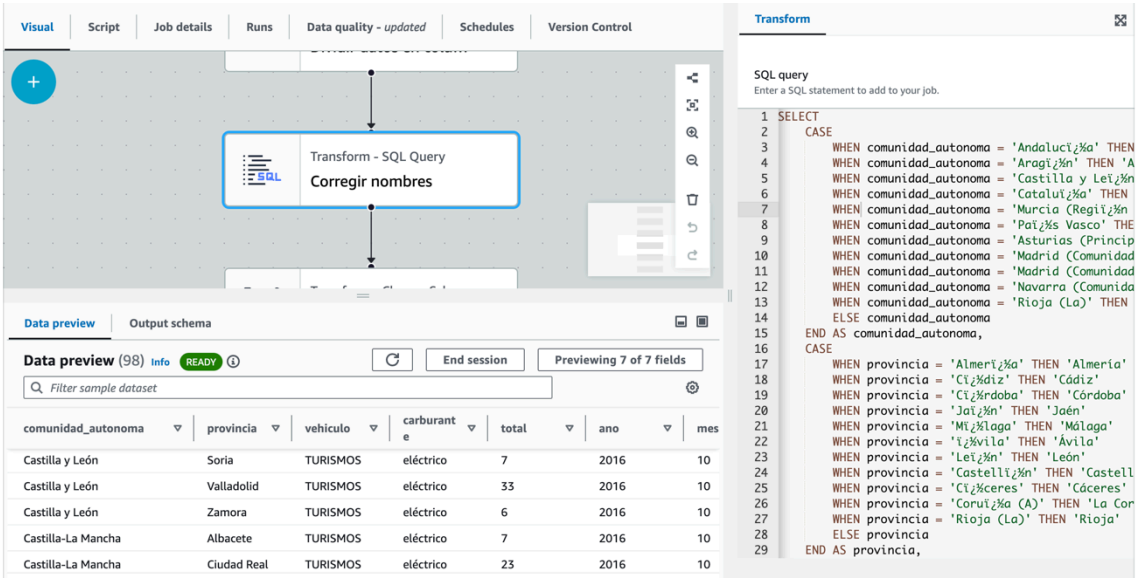


Figura 16: Transformación de datos

5. Carga de datos en S3: Una vez que los datos han sido transformados y se asegura que tienen la estructura adecuada, se procede a realizar la transferencia y almacenamiento de los datos en S3 bucket, “clean-data-ve”.

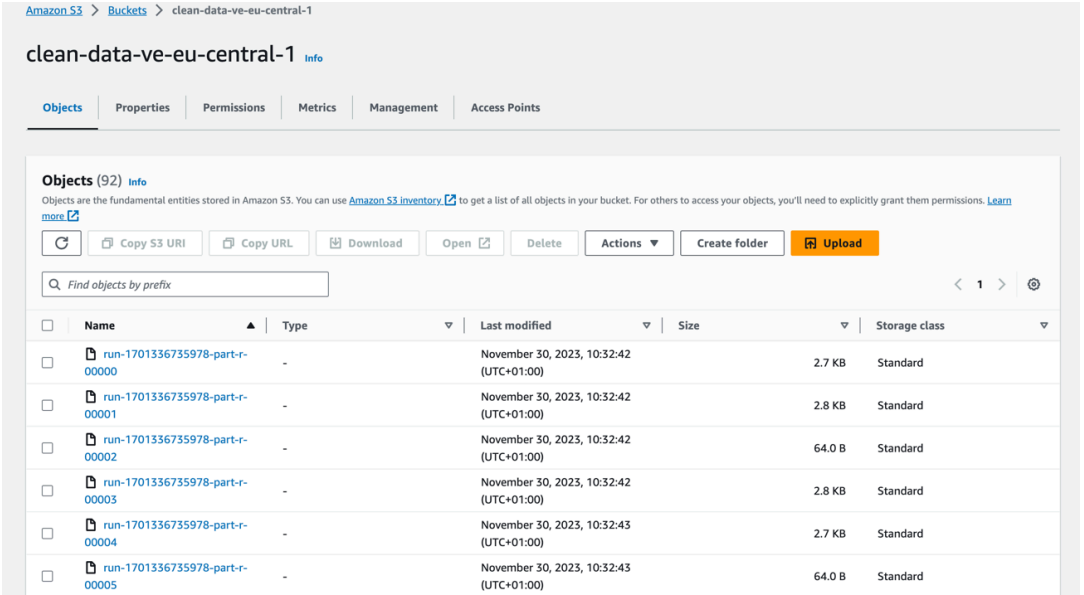


Figura 17: Carga de datos en S3 bucket

Una vez completadas las tareas de transformación con AWS Glue, los datos preparados y transformados podrán ser utilizados en los modelos de predicción. Este proceso de transformación es fundamental para garantizar que los datos sean coherentes, completos y adecuados para su posterior análisis y la obtención de resultados precisos.

4.4.2.3. Integración de modelos de predicción

Se ha implementado un componente de predicción utilizando Python, que integra dos algoritmos de aprendizaje automático: Prophet y regresión lineal. Estos algoritmos están entrenados previamente para predecir la demanda futura de coches eléctricos en España. Recibe los datos de entrada y devuelve las predicciones correspondientes.

El algoritmo Prophet se basa en modelos de series temporales y es especialmente adecuado para identificar tendencias y patrones estacionales en los datos históricos de demanda. Funciona mejor con series temporales que capturan efectos estacionales y varias temporadas de datos históricos. Resiste muy bien ante la falta de datos y a los cambios en la tendencia, y por lo general es capaz de manejar de manera adecuada los valores atípicos. Sin embargo, cuando la muestra de datos presenta cierta aleatoriedad, el algoritmo puede tener una precisión ligeramente menor. Al tener en cuenta los patrones estacionales, así como las tendencias generales, se obtienen predicciones más precisas y confiables.

El algoritmo de regresión lineal se utiliza para establecer una relación lineal entre la variable independiente (año) y la variable dependiente (demanda de coches eléctricos). Este enfoque busca encontrar una línea recta que mejor se ajuste a los datos históricos. La variable independiente año representa el año en el que se registraron las ventas de coches eléctricos. Esta variable se utiliza para hacer la predicción de las ventas futuras. La variable dependiente, total_coches, representa las ventas totales de coches eléctricos y esta es la variable que queremos predecir.

Se incluyen gráficos que ilustran las predicciones realizadas por Prophet y regresión lineal:

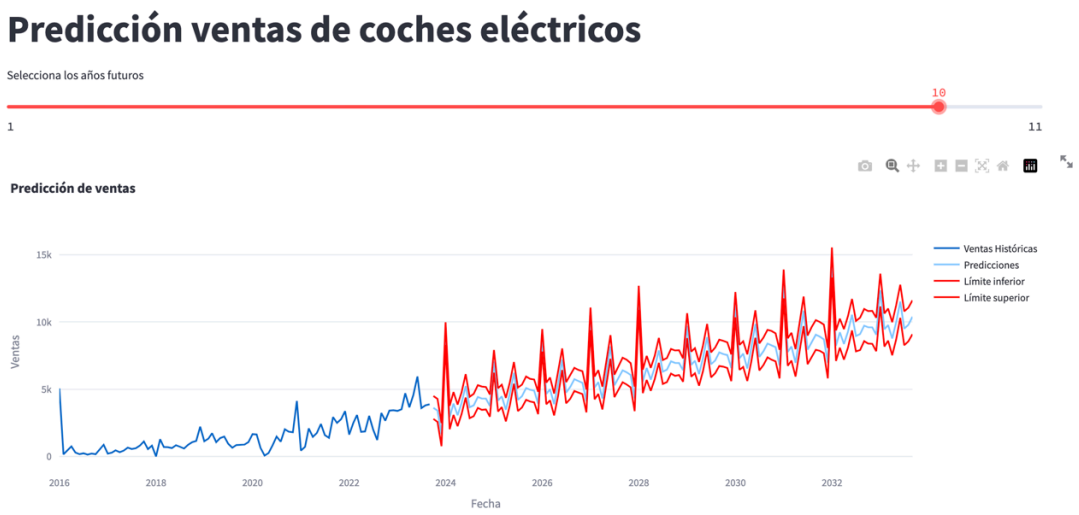


Figura 18: Proyección de ventas con algoritmo Prophet

Figura 19: Proyección de ventas con algoritmo de Regresión lineal

Para futuras investigaciones, se sugiere explorar otros algoritmos y técnicas que podrían complementar o mejorar los modelos existentes. Además, se podría considerar la opción de incluir más variables predictoras como por ejemplo la demografía para poder enriquecer la capacidad predictiva de los modelos.

4.4.3 Retos y lecciones aprendidas

El primero de los desafíos que me he encontrado al implementar el proyecto ha sido la recolección manual de los datos debido a la falta de una API para descargar los datos automáticamente de la DGT (Dirección General de Tráfico). Para superar este desafío, se han descargado los datos manualmente de la web de la DGT y posteriormente se ha desarrollado un script en Python como una alternativa para cargar los datos automáticamente en el S3 bucket de aws.

El siguiente desafío ha sido la falta de datos en el año 2018 del mes de enero. Dado que no hay datos disponibles en este período, se toma la decisión de utilizar los datos del mes anterior para llenar este vacío temporal en la información.

4.5. Despliegue

Durante el proceso de despliegue inicial en Amazon Web Services (AWS), se crea un entorno en la nube que incluye la configuración de un S3 bucket, que actúa como un repositorio centralizado para almacenar los datos relacionados con la demanda de coches eléctricos.

La elección de un almacenamiento en la nube ofrece significativas ventajas. En primer lugar, ofrece una inversión económica baja en comparación con las infraestructuras

tradicionales on-premise, que suelen ser más costosas y requieren un mantenimiento y gestión continuos.

Además, el almacenamiento en la nube ofrece la flexibilidad de adaptarse rápidamente a las cambiantes necesidades de almacenamiento y procesamiento de datos. Puede escalar verticalmente para manejar el aumento en la demanda de almacenamiento, permitiendo almacenar grandes volúmenes de información sin problemas.

El despliegue en AWS garantiza que el entorno se encuentre disponible y accesible en la nube, permitiendo que Streamlit pueda interactuar con el S3 bucket y los datos almacenados en él de forma eficiente y segura.

Una vez finalizada la implementación de la aplicación, se procedió al despliegue utilizando la versión gratuita de Streamlit. Esta opción de despliegue gratuito proporcionada por Streamlit ofrece la posibilidad de alojar la aplicación en un entorno en la nube accesible para los usuarios.

Aunque se ha utilizado la versión gratuita, proporciona funcionalidades y características suficientes para garantizar la disponibilidad y rendimiento de la aplicación. También existen opciones de pago con capacidades adicionales pero la versión gratuita permite compartir y utilizar la aplicación sin incurrir en ningún coste adicional.

El despliegue llevado a cabo con Streamlit permite a los usuarios acceder a la aplicación de manera sencilla a través del navegador web, sin la necesidad de realizar instalaciones o configuraciones complejas. Esto proporciona una experiencia conveniente para los usuarios finales, quienes pueden interactuar con la aplicación y beneficiarse de las funcionalidades.

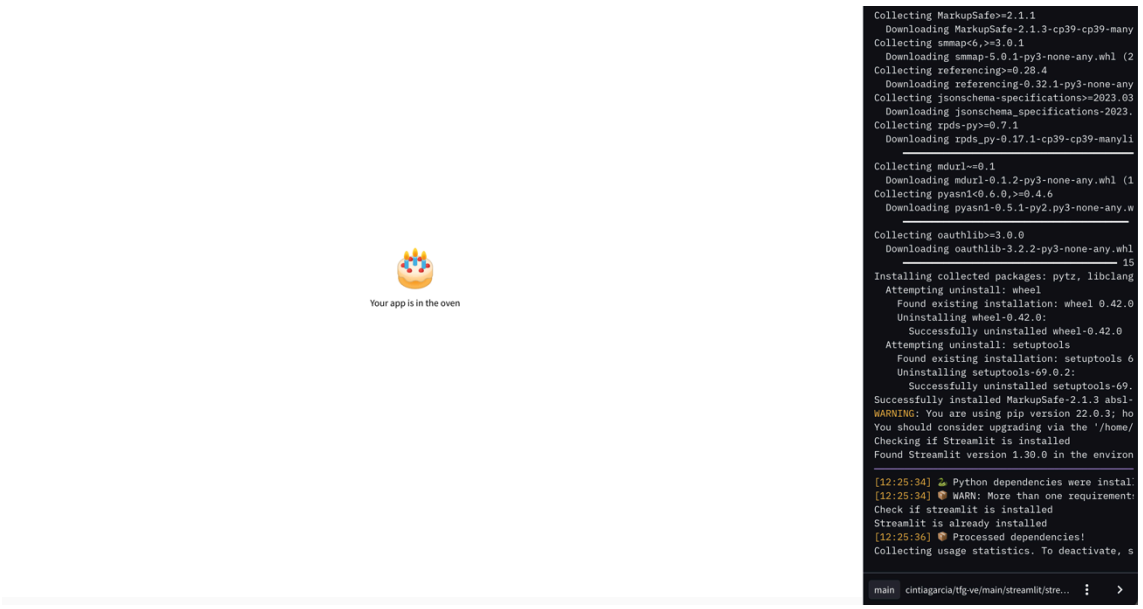


Figura 20: Despliegue realizado con Streamlit

4.6. Rendimiento

Se aborda el análisis del rendimiento en el contexto de almacenamiento en el S3 de Amazon Web Services. El monitoreo y la evaluación del rendimiento son aspectos fundamentales para garantizar la disponibilidad, confiabilidad y eficiencia de las aplicaciones y servicios alojados en S3. En este apartado me enfocaré en las métricas clave: Almacenamiento, Peticiones al S3 bucket, Bytes descargados, Errores 4xx y 5xx, Latencia del primer Byte. A través de su seguimiento se podrá identificar posibles problemas y tomar medidas proactivas para optimizar los recursos y mejorar la experiencia de usuario.

4.6.1 Almacenamiento

La métrica del almacenamiento consiste en el espacio total utilizado en el bucket de S3. Es crucial monitorear el crecimiento del almacenamiento para asegurarse de que existe suficiente capacidad disponible para satisfacer las necesidades futuras y actuales. El seguimiento de esta métrica permite identificar patrones de crecimiento, planificar expansiones y tomar decisiones informadas sobre escalar el sistema.

Actualmente, en el S3 bucket se ha utilizado un espacio de almacenamiento de 181.9 KB, como se puede observar en la siguiente gráfica.

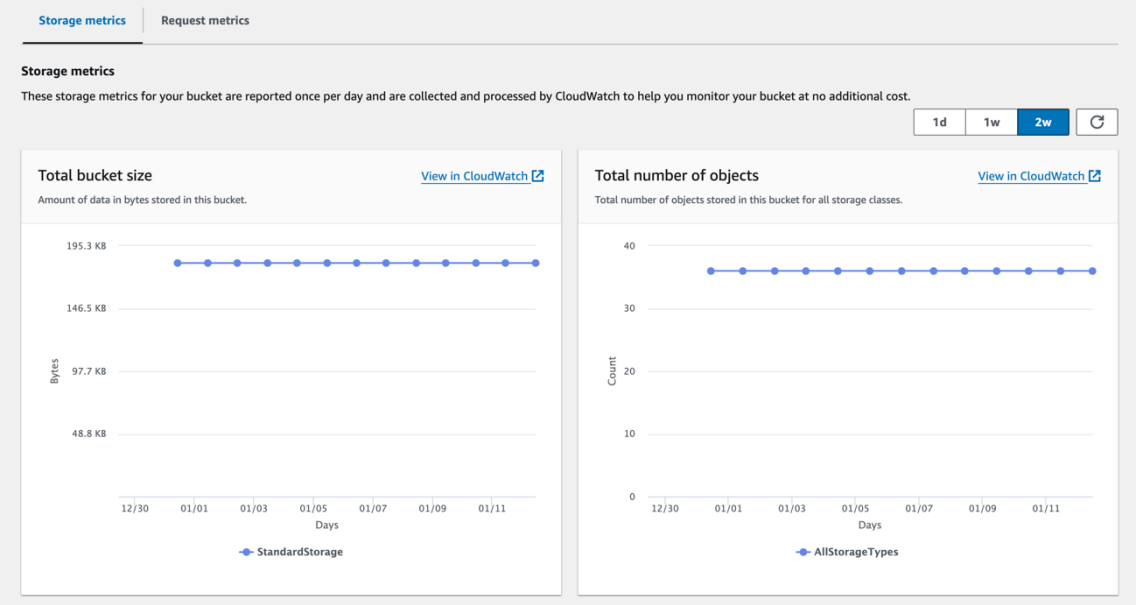


Figura 21: Almacenamiento en S3

4.6.2 Peticiones al S3 bucket

El número de peticiones al bucket de S3 es una métrica crucial para poder ser capaz de evaluar la carga y tráfico que el servicio recibe. Registrar y capturar el número de

peticiones permite identificar patrones de uso y evaluar la demanda del sistema. Esta petición es útil para optimizar la configuración y escalado del bucket, asegurando una respuesta eficiente y adecuada a las solicitudes de los usuarios.

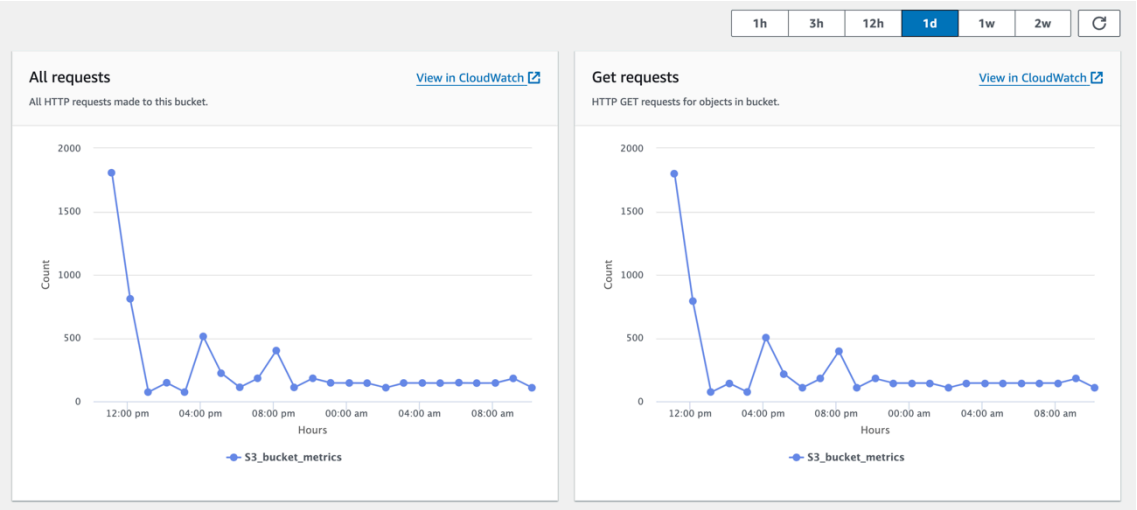


Figura 22: Peticiones realizadas al S3

4.6.3 Bytes descargados

En esta métrica se va a proporcionar información sobre la transferencia de datos. Monitorear los bytes descargados permite evaluar la carga de trabajo en el servicio, identificar picos de tráfico y planificar la capacidad de almacenamiento y ancho de banda necesario.

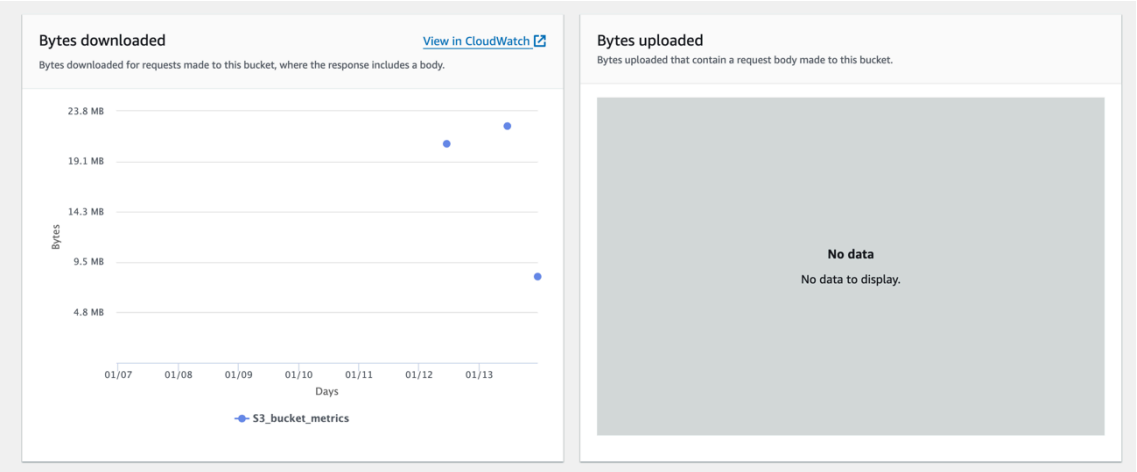


Figura 23: Bytes descargados

4.6.4 Errores 4xx y 5xx

Es importante hacer un seguimiento de los errores 4xx y 5xx para identificar posibles problemas en las solicitudes y respuestas del servidor. Los errores 4xx se basan en peticiones incorrectas del lado del cliente, como puede ser “401 Autorization Required”, “403 Forbidden” o “404 Not found”, indicando que la solicitud realizado por el cliente no ha podido ser procesada correctamente. Los errores 5xx son errores por parte del servidor, como puede ser “500 Internal Server Error”, que indican que hubo un problema con el servidor al procesar la solicitud.

Al analizar la frecuencia y el tipo de errores 4xx y 5xx, es posible identificar patrones problemáticos y tomar las medidas correctivas adecuadas para mejorar la disponibilidad y experiencia de usuario.

Actualmente, en el bucket S3 no se han registrado errores 4xx ni tampoco 5xx, como se puede apreciar en la siguiente gráfica:

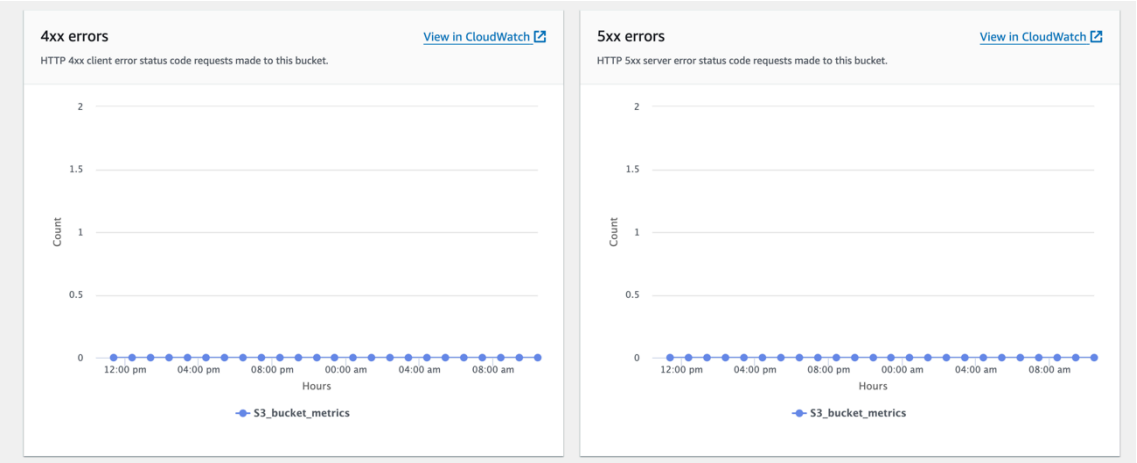


Figura 24: Errores 4xx y 5xx

4.6.5 Latencia del primer Byte

La latencia del primer byte registra el tiempo que transcurre desde el momento que se establece la conexión entre el servidor y el cliente, hasta que el primer byte de datos llega al cliente. Es una métrica esencial, ya que indica la velocidad de respuesta del servidor. Una baja latencia del primer byte generalmente se asocia con un rendimiento rápido del servidor. Al monitorear esta métrica es posible detectar problemas de rendimiento o posibles cuellos de botella en el rendimiento del servidor.

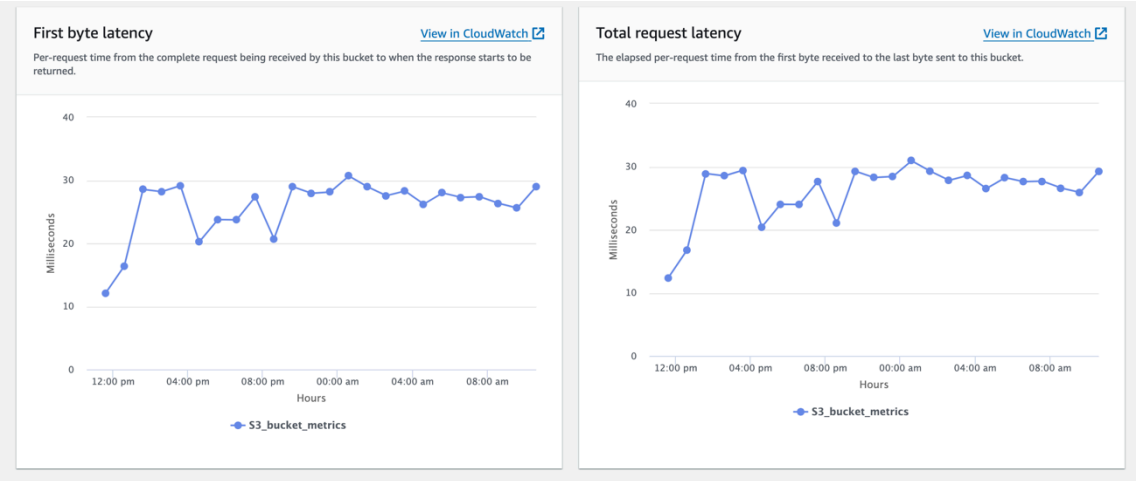


Figura 25: Latencia del primer byte

4.7. Mantenimiento

El mantenimiento es una fase fundamental para garantizar el funcionamiento continuo de la aplicación. Asimismo, se ha establecido un plan de actualizaciones periódicas para agregar nuevas funcionalidades y mejoras a la aplicación, con el objetivo de satisfacer las necesidades cambiantes de los usuarios y poder mantenerla al día con las últimas tecnologías y tendencias en el campo de la predicción de la demanda de coches eléctricos.

En cuanto al monitoreo y gestión de incidencias, se ha utilizado CloudWatch, un servicio de monitorización de AWS. CloudWatch permite recopilar y visualizar métricas relevantes, generar alarmas y obtener información detallada sobre el rendimiento de los recursos en AWS. Mediante la configuración adecuada de CloudWatch, se pueden supervisar métricas clave con el tiempo de respuesta de la aplicación, la utilización de recursos y demás indicadores necesarios.

Al utilizar CloudWatch, se pueden establecer umbrales y alarmas para recibir notificaciones en caso de que los límites definidos hayan sido superados. Esto permite identificar y ser capaz de solucionar de manera proactiva posibles incidencias o problemas de rendimiento antes de que afecten negativamente a la aplicación. Este tipo de acciones contribuye a mantener un funcionamiento óptimo de la aplicación y brindar una experiencia satisfactoria a los usuarios.

Capítulo 5: Pruebas y Evaluación

5.1. Pruebas

Las pruebas de aceptación son una parte fundamental cuando se ha desarrollado una aplicación web. Estas pruebas van a permitir verificar que la aplicación cumple con los requisitos y expectativas del usuario antes de la implementación final. Se han desarrollado las siguientes pruebas de aceptación:

5.1.1 CP-1: Visualización de datos históricos

Visualización de datos históricos	**CP-1**
Descripción: La prueba de visualización de datos históricos tiene como objetivo verificar que la funcionalidad de la pestaña "Datos históricos" de la web app "VE Predicción" está cumpliendo con los requisitos establecidos. En esta prueba, se busca confirmar que los gráficos con los datos históricos de ventas de coches eléctricos se muestran correctamente, abarcando el período desde 2016 hasta 2023.	
Prerrequisitos: ● Tener acceso a la aplicación web "VE Predicción" ● Contar con datos históricos de ventas de coches eléctricos desde 2016 hasta 2023 en la base de datos de la web app.	
Pasos: 1. Acceder a la pestaña "Datos históricos" 2. Verificar que se muestran los gráficos con datos históricos de ventas de coches eléctricos desde 2016 hasta 2023 3. Filtrar por Comunidad Autónoma y seleccionar un año para verificar que los datos se actualizan correctamente	
Resultado esperado: ● Los gráficos con los datos históricos de ventas de coches eléctricos se cargan correctamente ● Los gráficos muestran los datos desde el 2016 hasta 2023	

• Al aplicar el filtro por Comunidad Autónoma y seleccionar un año, los gráficos se actualizan y muestran los datos correspondientes a la selección
Resultado obtenido: **Los gráficos se cargan correctamente y muestran los datos históricos de ventas de coches eléctricos desde 2016 hasta 2023. Al aplicar el filtro por Comunidad Autónoma y seleccionar un año, los gráficos se actualizan adecuadamente y muestran los datos correspondientes a la selección realizada.**

5.1.2 CP-2: Visualización de la predicción de ventas con el algoritmo Prophet

Visualización de predicción de ventas con Prophet	**CP-2**
Descripción: En esta prueba, se busca verificar la funcionalidad de la visualización de la predicción de ventas de coche eléctricos utilizando el algoritmo Prophet en la pestaña "Tendencias" de la aplicación web "VE Predicción"	
Prerrequisitos: • Tener acceso a la aplicación web "VE Predicción" • Contar con datos históricos de ventas de coches eléctricos desde 2016 hasta 2023 en la base de datos de la web app.	
Pasos: Acceder a la pestaña "Tendencias" Verificar que se muestran correctamente la predicción de ventas de coches eléctricos que utiliza el algoritmo Prophet Comparar la predicción con los datos históricos para evaluar la precisión del modelo	
Resultado esperado: • La visualización de la predicción de ventas de coches eléctricos utilizando el algoritmo Prophet se muestra correctamente en la pestaña "Tendencias" • La predicción debe reflejar de manera precisa la tendencia de ventas de coches eléctricos a lo largo del tiempo, considerando los datos históricos.	
Resultado obtenido: **La visualización de la predicción de ventas con el algoritmo Prophet se muestra correctamente en**	

la pestaña "Tendencias". La predicción se ajusta de manera precisa a los datos históricos, reflejando la tendencia de ventas de coches eléctricos a lo largo del tiempo.

5.1.3 CP-3: Visualización de la predicción de ventas con el algoritmo de Regresión Lineal

Visualización de predicción de ventas con Regresión Lineal	**CP-3**
Descripción: En la visualización de predicción de ventas con el modelo de Regresión Lineal se busca verificar la funcionalidad de la visualización de la predicción de ventas de coches eléctricos utilizando el algoritmo de Regresión Lineal en la pestaña "Tendencias" de la aplicación web "VE Predicción"	
Prerrequisitos: ● Tener acceso a la aplicación web "VE Predicción" ● Contar con datos históricos de ventas de coches eléctricos desde 2016 hasta 2023 en la base de datos de la web app.	
Pasos: 1. Acceder a la pestaña "Tendencias" 2. Verificar que se muestran a predicción de ventas de coches eléctricos utilizando el algoritmo de regresión lineal 3. Comparar la predicción con los datos históricos para evaluar la precisión del modelo	
Resultado esperado: ● La visualización de la predicción de ventas de coches eléctricos utilizando el algoritmo de regresión lineal se muestra correctamente en la pestaña "Tendencias" ● La predicción deber reflejar de manera precisa la tendencia de ventas de coches eléctricos a lo largo de 10 años, considerando los datos históricos	
Resultado obtenido: **La visualización de la predicción de ventas con el algoritmo de regresión lineal se muestra correctamente en la pestaña "Tendencias". La predicción se ajusta de manera precisa reflejando la**	

tendencia de ventas de coches eléctricos a lo largo del tiempo

5.1.4 CP-4: Prueba de navegación y funcionalidad del panel de navegación

Prueba de funcionalidad del panel de navegación	**CP-4**
Descripción: En esta prueba se busca verificar la funcionalidad del panel de navegación en la aplicación web "VE Predicción". El objetivo es comprobar que la barra de navegación lateral izquierda funciona correctamente y que se muestre el contenido correspondiente a cada una de ellas	
Prerrequisitos: ● Tener acceso a la aplicación web "VE Predicción"	
Pasos: 1. Acceder a cada una de las pestañas del panel de navegación (Home, Datos históricos, Tendencias) 2. Verificar que se muestra el contenido correspondiente a cada pestaña de forma adecuada 3. Comprobar que la navegación entre las pestañas funciona correctamente, es decir, al hacer clic en cada pestaña, se muestra el contenido respectivo sin problemas 4. Comprobar que al hacer clic en la "x" de la barra lateral izquierda de navegación, esta se cierre correctamente, ocultando la barra lateral y ampliando el espacio del contenido principal de la aplicación.	
Resultado esperado: ● Al acceder a la pestaña "Home", se debe mostrar el contenido de la página principal de la aplicación ● Al acceder a la pestaña "Datos históricos", se espera visualizar los gráficos y datos históricos de ventas de coches eléctricos ● Al acceder a la pestaña "Tendencias", se espera ver la predicción de ventas utilizando los algoritmos Prophet o Regresión Lineal ● La navegación entre las pestañas debe ser fluida y sin errores ● Al hacer clic en la "x" de la barra lateral izquierda de navegación, esta se cierra correctamente	
Resultado obtenido:	

Al acceder a cada una de las pestañas del panel de navegación, se muestra el contenido correspondiente de forma correcta. La navegación entre las pestañas funciona sin problemas, permitiendo acceder al contenido deseado en cada pestaña. La barra lateral izquierda de navegación se cierra correctamente al presionar la "x".

5.1.5 CP-5: Prueba de usuarios objetivo:

Prueba de usuario objetivo	**CP-5**
Descripción: En esta prueba, se busca evaluar la experiencia de uso de un usuario objetivo en la aplicación web "VE Predicción". El objetivo es comprobar que el usuario puede acceder y utilizar la aplicación	
Prerrequisitos: ● Tener acceso a la aplicación web "VE Predicción"	
Pasos: 1. El usuario accede a la aplicación web "VE Predicción" 2. El usuario navega por las diferentes secciones y funcionalidades de la aplicación, como la visualización de datos históricos y las tendencias de ventas 3. El usuario realiza acciones específicas dentro de la aplicación, como seleccionar un rango de fechas, interactuar con los gráficos y visualizar los resultados de la predicción 4. Se registra la facilidad de uso, la comprensión de las funcionalidades y la satisfacción del usuario durante la prueba	
Resultado esperado: ● El usuario debería poder acceder a la aplicación sin problemas ● El usuario debería ser capaz de navegar por las diferentes secciones y funcionalidades de la aplicación de forma intuitiva ● El usuario debería poder realizar las acciones específicas de manera clara y sin dificultades ● El usuario debería sentirse satisfecho con la experiencia de uso de la aplicación	
Resultado obtenido: **El usuario puede acceder a la aplicación web sin inconvenientes. Navegó por las diferentes secciones y funcionalidades de manera intuitiva y pudo realizar las acciones específicas sin**	

dificultades. El usuario se mostró satisfecho con la experiencia de uso de la aplicación.

5.2. Evaluación

En este apartado, se llevará a cabo una evaluación del sistema desarrollado para predecir las ventas de coches eléctricos, así como de las tecnologías en la nube utilizadas.

Como se planteó inicialmente en los objetivos del proyecto, el enfoque no se limitó únicamente al análisis de las ventas de vehículos eléctricos, sino también se tuvo en consideración la evaluación de diversas tecnologías en la nube y se exploraron las oportunidades que brindan para aprovechar datos públicos mediante la implementación de algoritmos de aprendizaje automático. Cabe destacar que esta solución es altamente versátil y adaptable, lo que permite su aplicabilidad en cualquier otro contexto, organización o empresa.

Durante el desarrollo del proyecto, se evaluaron diferentes tecnologías en la nube para determinar la más adecuada para la implementación de la web. Se consideraron proveedores de servicios en la nube como AWS, Google Cloud y Microsoft Azure. Después de un análisis exhaustivo, se decidió utilizar AWS debido a la amplia gama de servicios y su escalabilidad. Esta elección permite aprovechar las capacidades de la nube, como el almacenamiento y el procesamiento de datos. Además, el uso de un S3 bucket facilitó el acceso y la integración de los datos en el sistema de predicción.

Gracias a la infraestructura en la nube y los algoritmos de aprendizaje automático utilizados, esta solución es aplicable a diferentes escenarios. Por ejemplo, se puede adaptar para predecir la demanda de otros productos o servicios, o para realizar análisis de mercado en diversos sectores.

Es importante destacar que se ha implementado una cuidada interfaz de fácil uso para el usuario, permitiéndole interactuar con el sistema de manera intuitiva. Se ha diseñado una interfaz gráfica amigable que facilita la navegación y el acceso a las funcionalidades de la aplicación web.

Además, se ha añadido un chat interactivo donde el usuario puede interactuar directamente con el sistema. Mediante este chat, el usuario tiene la capacidad de hacer preguntas relacionadas con las ventas de coches eléctricos, obtener respuestas instantáneas y proporcionar retroalimentación. Esta funcionalidad mejora la experiencia del usuario al brindarle una forma más dinámica y conversacional para poder interactuar con el sistema.

La inclusión de esta interfaz de fácil uso y el chat interactivo no solo mejora la experiencia del usuario, sino también aumenta la usabilidad y accesibilidad de la solución. Estas características adicionales contribuyen a la adopción y aceptación del sistema, permitiendo a los usuarios obtener respuestas rápidas y relevantes a sus consultas relacionadas con las ventas de coches eléctricos.

Durante las pruebas de aceptación, se observó que había demoras significativas en la carga de datos. Para abordar este problema, se implementó un sistema de caché en el código con el objetivo de reducir los tiempos de espera. Se realizaron pruebas en un entorno local (localhost), donde se constató que el tiempo de carga se redujo a la mitad.

Sin embargo, al desplegar la solución con Streamlit y volver a probar la funcionalidad, se notó que el tiempo de espera para la carga de datos había aumentado nuevamente. Esto indica que se requieren mejoras adicionales en la infraestructura para optimizar el rendimiento.

Entre las posibles mejoras identificadas se encuentra la opción de cambiar a un servidor de pago con mayores capacidades y recursos para agilizar la carga de datos. Otra alternativa sería la creación de una base de datos que permita una carga más eficiente de los datos utilizados en el sistema.

Estas mejoras potenciales contribuirían a optimizar el tiempo de espera de carga de datos, mejorando así la experiencia del usuario y garantizando un rendimiento óptimo de la solución en su implementación final.

Capítulo 6: Resultados y Conclusiones

6.1. Conclusiones en relación con los objetivos

En este proyecto se ha alcanzado el objetivo principal de desarrollar un sistema de predicción de ventas de coches eléctricos, brindando a las partes interesadas herramientas y modelos predictivos para anticipar las tendencias del mercado y tomar las decisiones estratégicas. A través de la implementación de una infraestructura en la nube y el uso de algoritmos de aprendizaje automático e inteligencia artificial, se ha logrado generar pronósticos precisos de la demanda futura de vehículos eléctricos.

Además, se ha diseñado una interfaz intuitiva y accesible, que permite a los usuarios interactuar con los resultados de las predicciones, acceder a informes y gráficos generados, así como ajustar parámetros de entrada y simular escenarios hipotéticos. La incorporación de una funcionalidad de chat ha enriquecido la experiencia del usuario al brindar respuestas a preguntas sobre los datos históricos.

Durante el desarrollo de este proyecto, se ha contado con la contribución de diversos perfiles profesionales: ingeniero de datos, analista de datos, ingeniero de inteligencia artificial y programador fronted. La labor desempeñada por estos roles ha sido fundamental para llevar a cabo la recolección y procesamiento de datos relevantes, implementar algoritmos de predicción y desarrollar la interfaz de usuario de manera exitosa.

No obstante, es crucial resaltar que se ha enfrentado un desafío significativo al intentar implementar los datos relacionados con las estaciones de carga de coches eléctricos debido a la falta de información pública disponible sobre este tema. Sería necesario invertir más tiempo en una investigación exhaustiva para explorar este ámbito de

manera más detallada y ofrecer una solución alternativa que satisfaga las necesidades de las partes interesadas.

A pesar de las dificultades encontradas en relación con los datos de estaciones de carga, se ha logrado cumplir los objetivos establecidos al inicio de este proyecto. Además, se ha agregado una nueva funcionalidad mediante el chat, que actualmente se encuentra en fase de prueba y se espera mejorar en el futuro. El trabajo conjunto de los diferentes perfiles profesionales ha sido clave para el éxito de este proyecto.

6.2. Conclusiones en relación con la planificación inicial

Durante la ejecución de este proyecto, ha sido necesario ajustar la planificación inicial debido a los desafíos que se han presentado. Uno de los desafíos inesperados fue la falta de una API que permitiera la descarga automática de los datos públicos. De haber existido esta API, el proceso habría sido menos manual, y se habría desarrollado un script en Python que, a través de una función lambda, realizará una solicitud diaria a la API y descargara automáticamente los datos, almacenados en un S3 bucket.

Después de enfrentar el desafío de la descarga manual, surgió otro obstáculo relacionado con el formato en el que se encontraban los datos públicos descargados. No eran adecuados para su análisis directo, lo que requirió realizar varias transformaciones para que los datos fueran utilizables.

En cuanto al desarrollo del frontend de la aplicación, se decidió utilizar Streamlit por su facilidad de uso en comparación con Django, que resulta más complejo. Sin embargo, trabajar con Streamlit también ha presentado algunas limitaciones que se han tenido en cuenta durante el proceso.

A lo largo del proyecto se ha tenido que adaptar la planificación inicial debido a los desafíos encontrados. La falta de una API automatizada para la descarga de datos y la necesidad de transformar el formato de los datos descargados han sido obstáculos significativos. A pesar de esto, se ha logrado avanzar en el desarrollo de la aplicación, utilizando Streamlit como plataforma frontend.

6.3. Trabajos futuros propuestos

En este proyecto, se han logrado avances significativos en la comprensión de la transición hacia la movilidad eléctrica y en el desarrollo de herramientas y análisis relacionados. Sin embargo, para seguir impulsando el progreso y mejorar la toma de decisiones en este ámbito, se proponen trabajos futuros que amplíen y enriquezcan aún más este proyecto.

En primer lugar, se sugiere la incorporación de datos adicionales para enriquecer el análisis. Esto incluye la incorporación de datos de estaciones de carga de vehículos eléctricos, así como datos demográficos relevantes. Al combinar estas variables con las predicciones de ventas de coches eléctricos, se podrían obtener conocimientos más profundos sobre las necesidades de infraestructura de carga en diferentes áreas y evaluar el progreso hacia los objetivos establecidos.

Además, se propone comparar las ventas de coches de combustión y coches eléctricos a lo largo de los años. Esto permitiría visualizar las tendencias de mercado, identificar patrones de adopción y evaluar el impacto de la movilidad eléctrica en la industria automotriz.

Otro aspecto a considerar es la automatización del asistente de chat existente. Esto implicaría establecer una integración fluida entre el S3 bucket de datos y el asistente

basado en el lenguaje natural, aprovechando la capacidad de respuesta y procesamiento de lenguaje natural de la herramienta. Esto permitiría obtener respuestas rápidas y precisas a preguntas relacionadas con los datos recopilados y realizar tareas adicionales basadas en el lenguaje natural.

Otra mejora adicional a considerar consistiría en emplear un algoritmo de redes neuronales para el análisis. Así como la posterior comparación con otros algoritmos ya existentes.

En resumen, los trabajos futuros propuestos buscan ampliar y mejorar el proyecto actual, incorporando datos adicionales, realizando comparativas relevantes y automatizando el asistente de chat existente. Estas mejoras permitirán un análisis más completo y una experiencia interactiva para los usuarios, impulsando aún más el conocimiento y la toma de decisiones en el campo de la movilidad eléctrica.

Bibliografía

[1] AWS CDK Workshop: AWS CDK Intro Workshop. Retrieved January 14, 2024, from https://cdkworkshop.com/

[2] Arrastia, D., & Assalve, A. (2021, October 27). ¿Qué frena a los españoles a comprar un coche eléctrico? *La Vanguardia*. https://www.lavanguardia.com/motor/20211027/7818499/dudas-compra-coche-electrico-espanoles-cochesnet-brl.html

[3] Arriaga, L. (n.d.). *¿Qué es la metodología Kanban y cómo implementarla?* INESEM Business School. Retrieved January 14, 2024, from https://www.inesem.es/revistadigital/gestion-empresarial/kanban-el-metodo-para-desarrollar-proyectos-de-exito/

[4] *Cada vez tenemos más puntos de carga para coches eléctricos. No son suficientes, ni tienen la potencia necesaria*. (2023, August 13). Motorpasión. Retrieved January 14, 2024, from https://www.motorpasion.com/futuro-movimiento/cada-vez-tenemos-puntos-carga-para-coches-electricos-no-suficientes-tienen-potencia-necesaria

[5] *Connect Streamlit to AWS S3 - Streamlit Docs*. (n.d.). Streamlit documentation. Retrieved January 14, 2024, from https://docs.streamlit.io/knowledge-base/tutorials/databases/aws-s3

[6] *¿En qué consiste la prohibición de los coches de combustión en 2035?* (2023, July 31). RACE. Retrieved January 14, 2024, from https://www.race.es/prohibicion-coches-diesel

[7] *Estudio | Infraestructura de recarga para vehículos eléctricos en España*. (n.d.). ECODES. Retrieved January 14, 2024, from https://ecodes.org/hacemos/cambio-climatico/incidencia-en-politicas-publicas/estudio-infraestructura-de-recarga-para-vehiculos-electricos-en-espana

[8] *REGLAMENTO (UE) 2016/ 679 DEL PARLAMENTO EUROPEO Y DEL CONSEJO - de 27 de abril de 2016 - relativo a la prot*. (2016, April 27).

BOE.es. Retrieved January 14, 2024, from https://www.boe.es/doue/2016/119/L00001-00088.pdf

[9] Rodríguez, F. (2022, November 30). *¿Qué es TypeScript?* KeepCoding. Retrieved January 14, 2024, from https://keepcoding.io/blog/typescript/

[10] *Solo el 7% de los compradores de coche apuesta por un eléctrico*. (2021, June 3). Coches.net. Retrieved January 14, 2024, from https://www.coches.net/blog-profesionales/solo-el-7-de-los-compradores-de-coche-apuesta-por-un-electrico/

[11] *Un informe pronostica que en 2030 habrá casi 27 millones de vehículos eléctricos*. (2019, July 8). esmartcity. Retrieved January 14, 2024, from https://www.esmartcity.es/2019/07/08/informe-pronostica-2030-habra-casi-27-millones-vehiculos-electricos

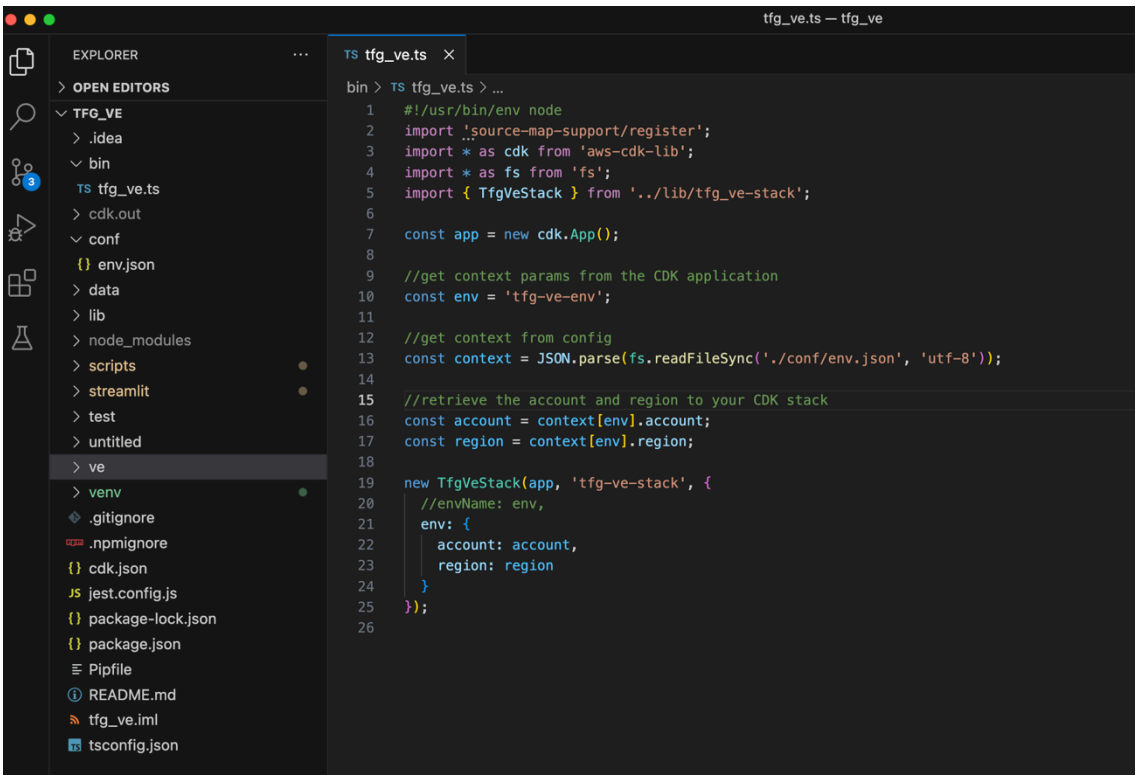
[12] "Portal estadístico." *Sede Electrónica de la DGT.*, https://sedeapl.dgt.gob.es/WEB_IEST_CONSULTA/subcategoria.faces. Accessed 14 January 2024.

Anexos

Anexo A1. Tecnologías utilizadas

AWS CDK (Cloud Development Kit): El AWS CDK es un marco de desarrollo de infraestructura como código que permite crear y administrar recursos de AWS utilizando lenguajes de programación. El proyecto se ha implementado utilizando Typescript como lenguaje de programación para definir la infraestructura en AWS CDK.

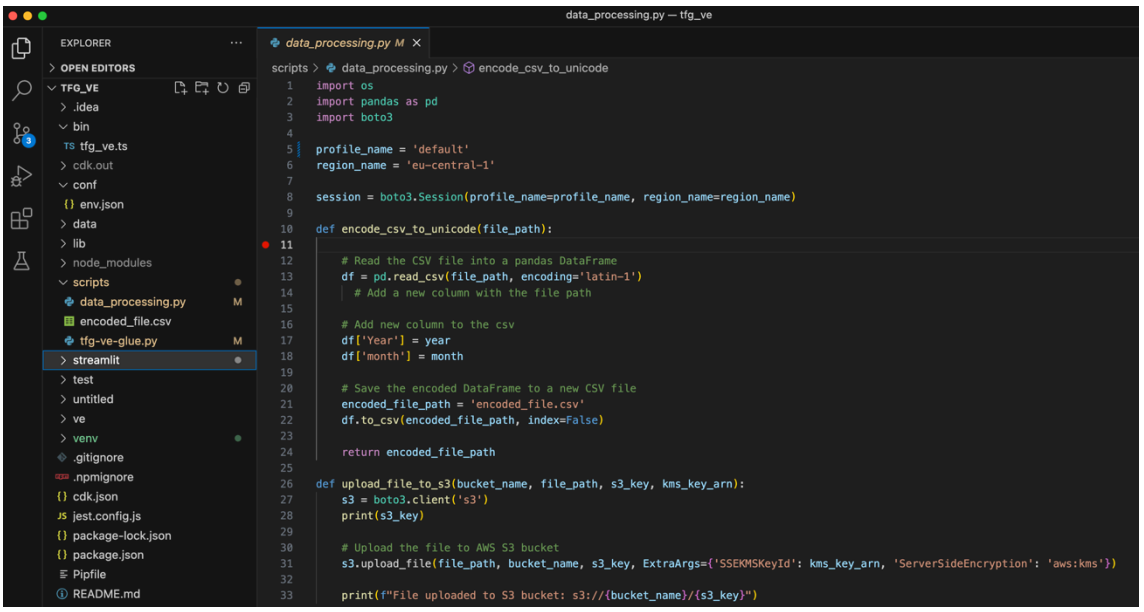
Además, como parte de la preparación y familiarización con el uso de AWS CDK en TypeScript he realizado un AWS workshop específico. Este taller me proporcionó un entorno de aprendizaje práctico y guiado, donde pude explorar los conceptos clave de AWS CDK, llevando a cabo ejercicios para construir y desplegar infraestructuras en la nube utilizando TypeScript como lenguaje de programación. El taller ha resultado fundamental para adquirir los conocimientos y habilidades necesarias para aprovechar al máximo en el desarrollo del proyecto.



TypeScript para definir infraestructura en AWS CDK

TypeScript: Es un lenguaje de programación de código abierto desarrollado por Microsoft. Es un superset de JavaScript, lo que significa que es compatible con la sintaxis y la semántica de JavaScript y lo amplía al agregar tipos estáticos opcionales y características adicionales. Permite detectar errores en tiempo de compilación en lugar de descubrir los errores en tiempo de ejecución.

Python: Es el lenguaje de programación ampliamente utilizado que se caracteriza por su simplicidad y legibilidad. En este proyecto se han utilizado scripts de Python para poder cargar los datos automáticamente al s3 bucket en aws, así como para proceder a la limpieza y transformación de los datos y por último proceder a la visualización de los datos a través de Streamlit.



Script de Python para cargar los datos en s3 bucket

SQL: Structured Query Language, es un lenguaje de consulta utilizado para administrar y manipular bases de datos relacionales. En el proyecto el lenguaje SQL, se ha utilizado para definir las transformaciones y consultas de datos en el proceso de ETL (Extract, Transform, Load). Se han incluido consultas de SQL en el archivo de aws glue en Python, donde se han realizado todas las fases de transformación de los datos.

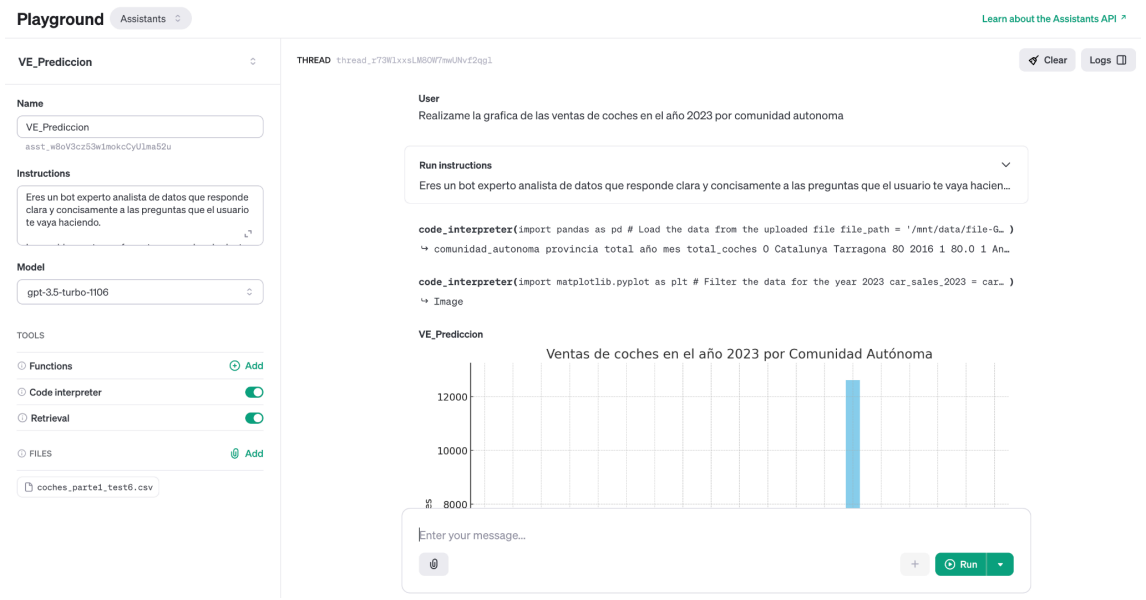
Streamlit: Es un marco de desarrollo de aplicaciones web en Python que facilita la creación de interfaces de usuario interactivas para mostrar y visualizar datos. No se requiere experiencia en front-end para poder utilizar este marco

Anexo A2. Incorporación de chat basado en LLM

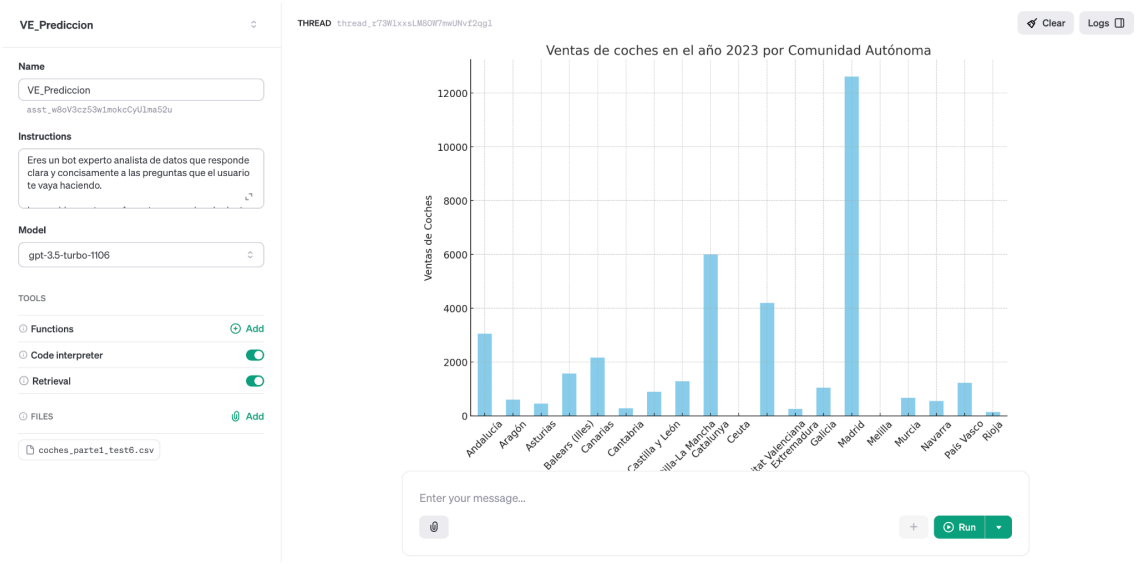
Incorporación de la base de conocimiento en el modelo LLM (Lenguaje y modelos de Lenguaje) de inteligencia artificial. Esta base de conocimiento permite al modelo acceder a la información actualizada y relevante sobre los datos históricos de ventas de coches eléctricos, lo que mejorará su capacidad para responder preguntas y ofrecer soluciones precisas.

Al agregar la función de chat de la aplicación web, se mejora significativamente la experiencia de usuario al permitirle interactuar y realizar preguntas específicas sobre las ventas de coches eléctricos en España.

Para implementar esta mejora, se ha desarrollado un asistente en la interfaz de openAI. Este asistente ha sido entrenado con un archivo específico y se le han proporcionado instrucciones detalladas para responder como un experto analista.

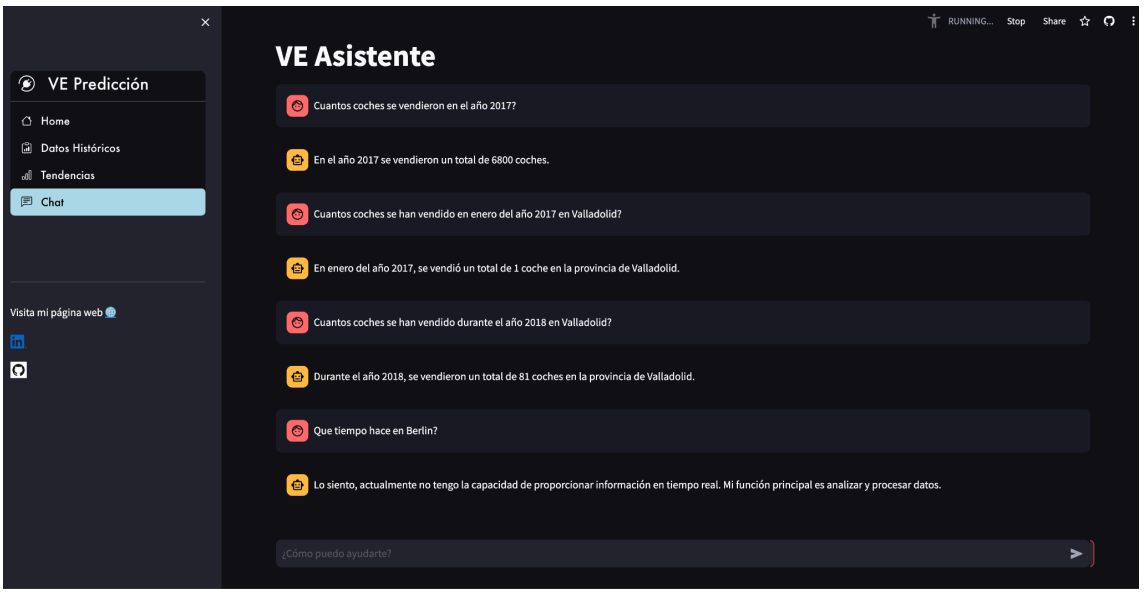


Asistente VE predicción creado en openAI



Asistente openAI realizando la gráfica de las ventas totales del año 2023

A continuación, se ilustra cómo el asistente de OpenAI es capaz de responder de manera precisa a las preguntas formuladas por el usuario, limitándose únicamente a aquellas que están relacionadas con el contenido del archivo.

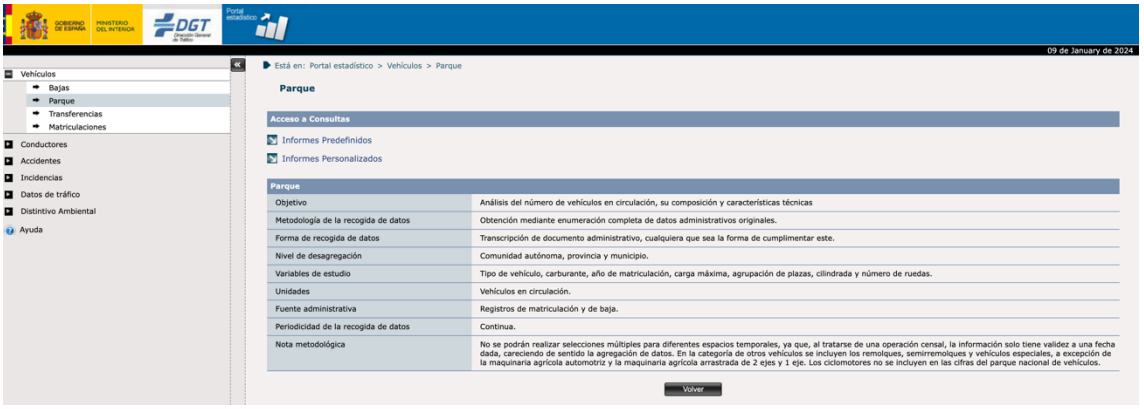


Asistente openAI contestando a las preguntas

Anexo A3. Extracción de datos de la web de la DGT

La extracción de los datos que se van a utilizar para el desarrollo de esta web app se encuentra en la página web del Portal Estadístico de la Dirección General de Tráfico. (Portal Estadístico DGT), específicamente dentro de la categoría 'Vehículos', subcategoría 'Parque' y la sección de 'Informes Predefinidos'.

Portal Estadístico de la DGT

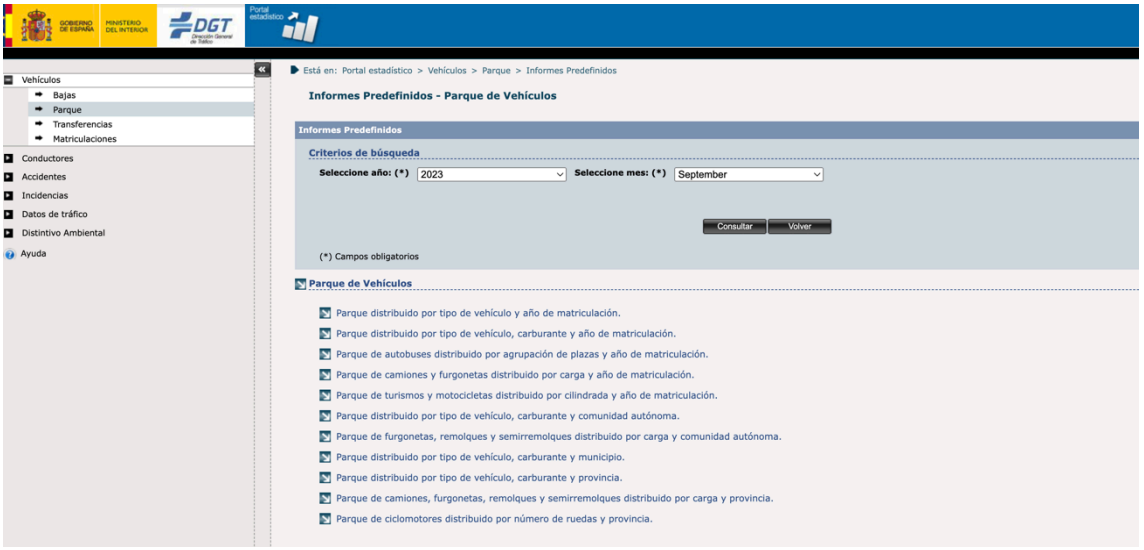


Portal Estadístico de la DGT de la categoría 'Vehículos'

Portal Estadístico de la DGT selección de Informes Predefinidos

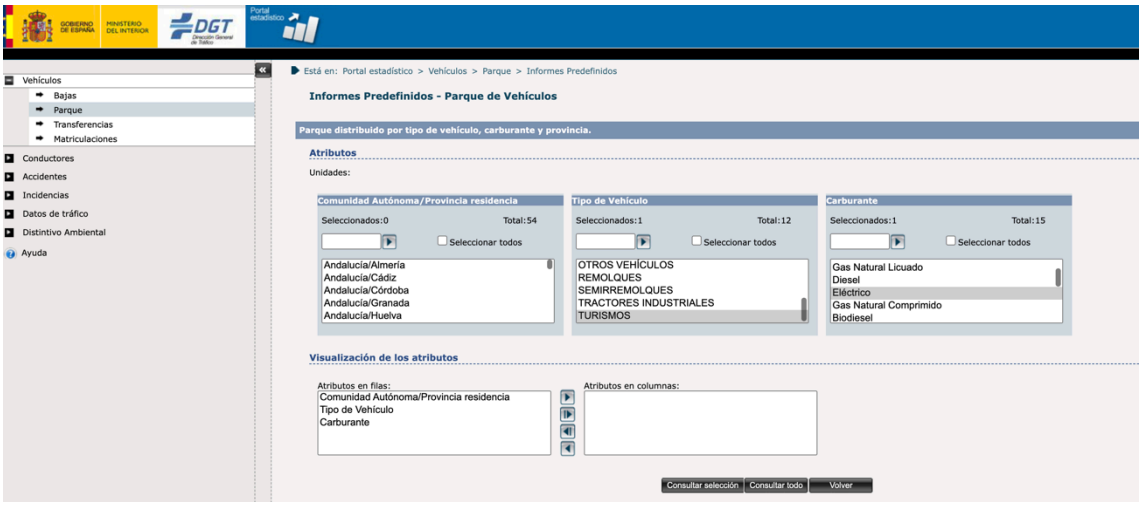
Dentro de la selección de "Informe Predefinidos – Parque de Vehículos", es necesario seleccionar el año y el mes deseado para realizar la consulta.

Optamos por seleccionar la opción: "Parque distribuido por tipo de vehículo, carburante y provincia", como los datos más apropiados para el desarrollo de la aplicación, considerando todas las opciones disponibles.



Portal Estadístico de la DGT selección de 'Informes Predefinidos' con diversas opciones disponibles

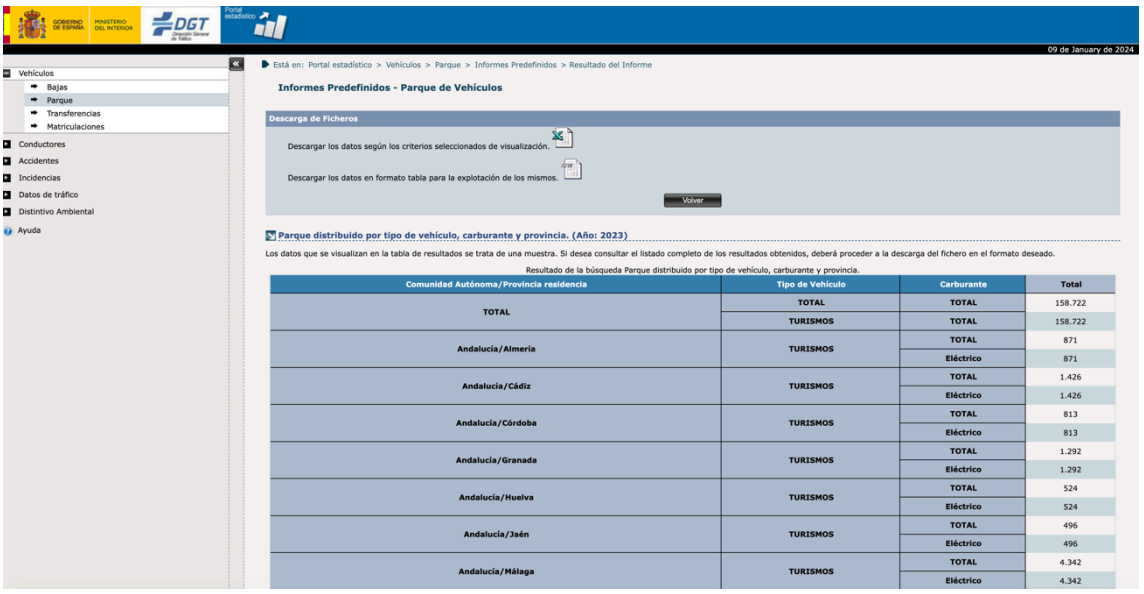
Seleccionamos todas las Comunidades Autónomas, mientras que para el tipo de vehículo seleccionamos ‘turismos’ y para el carburante elegimos ‘eléctrico’.



Portal Estadístico de la DGT selección de ‘Parque distribuido por tipo de vehículo, carburante y provincia’

Los datos se descargan del Portal Estadístico de la DGT manualmente en un archivo CSV. Esto implica que se realiza el proceso de descarga de manera manual, donde el usuario selecciona la opción de descarga y guarda los datos en un archivo en formato CSV o en formato EXCEL.

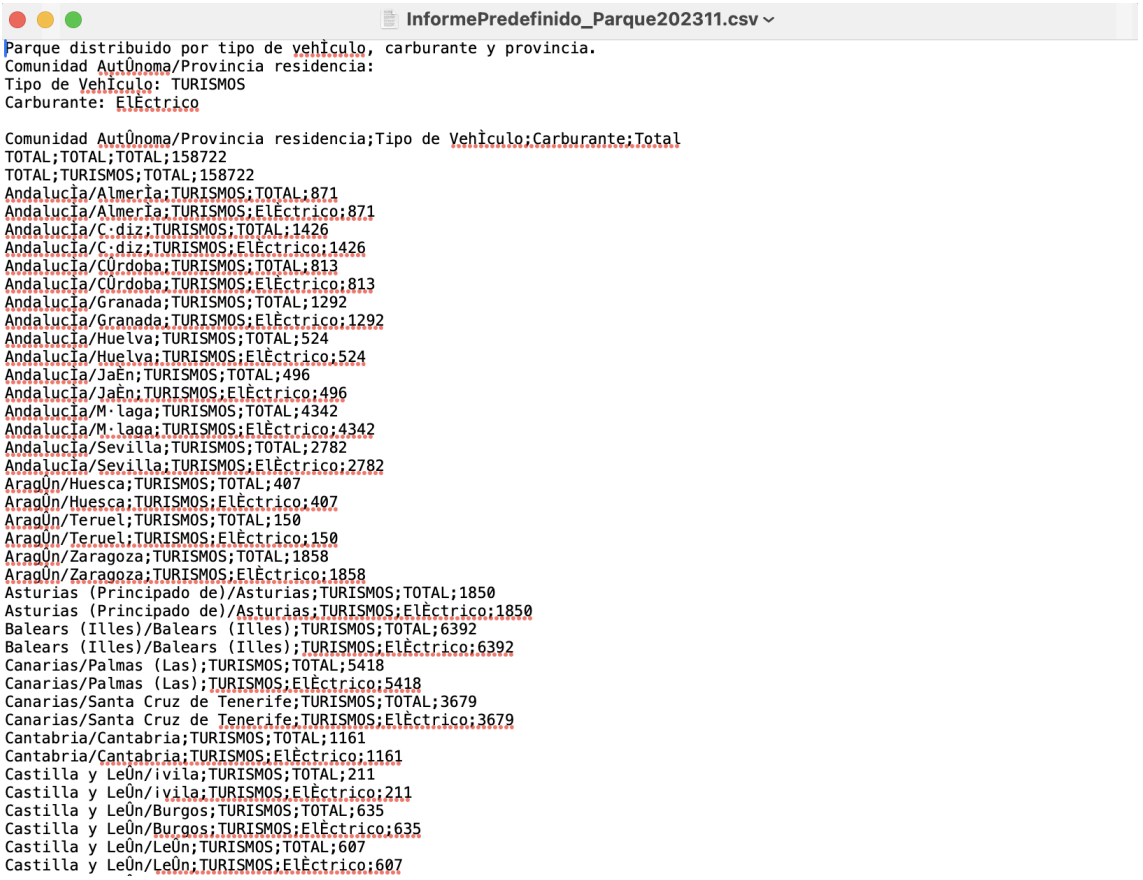
En mi caso, he optado por seleccionar la descarga en formato CSV para poder transformar los datos posteriormente. Esta elección me permite realizar manipulaciones y análisis en los datos descargados, ya que el formato CSV es ampliamente compatible con diversas herramientas y lenguajes de programación.



Se muestra una figura del Portal estadístico de la DGT, para el proceso de descarga de ficheros.

A continuación se presenta el fichero descargado. Como se puede observar, este se encuentra en formato CSV. Sin embargo, antes de utilizar los datos, será necesario eliminar ciertas filas que no resultan útiles, así como eliminar la línea vacía.

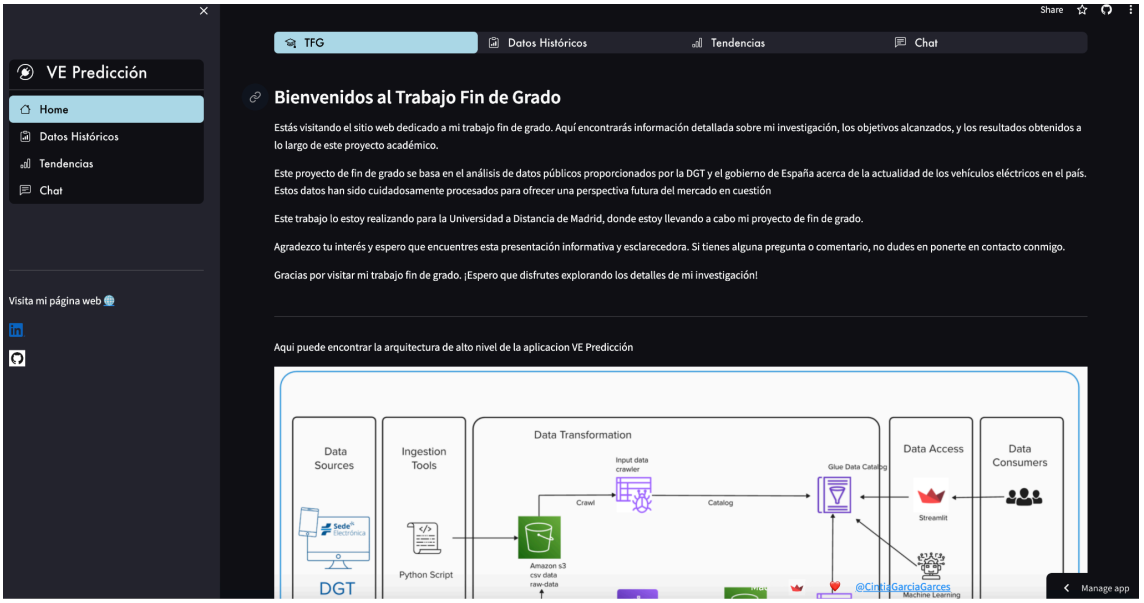
Además, es importante tener en cuenta que los datos del fichero no incluyen una columna adicional que muestre el año y el mes. Por lo tanto, será necesario realizar modificaciones adicionales en el fichero para incorporar esta información relevante.



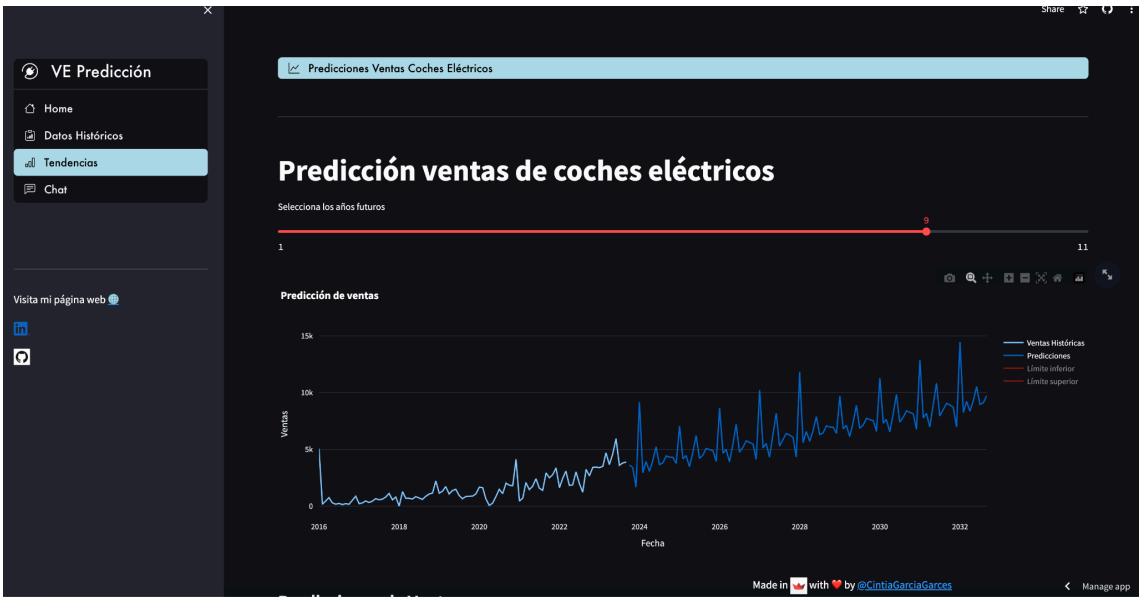
Se muestra el fichero descargado en formato CSV

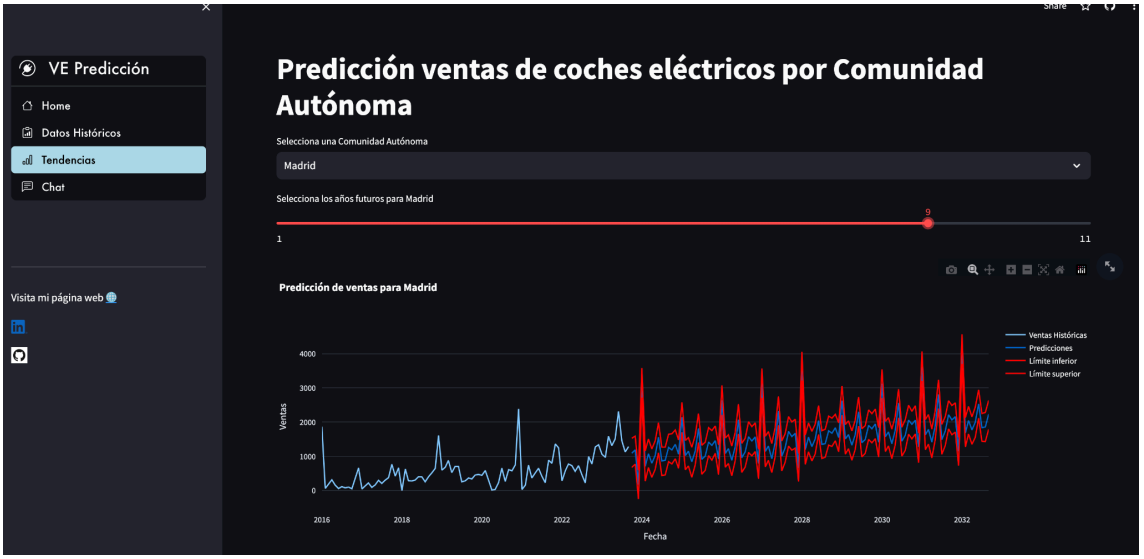
Anexo A4. Aplicación Web | VE Predicción

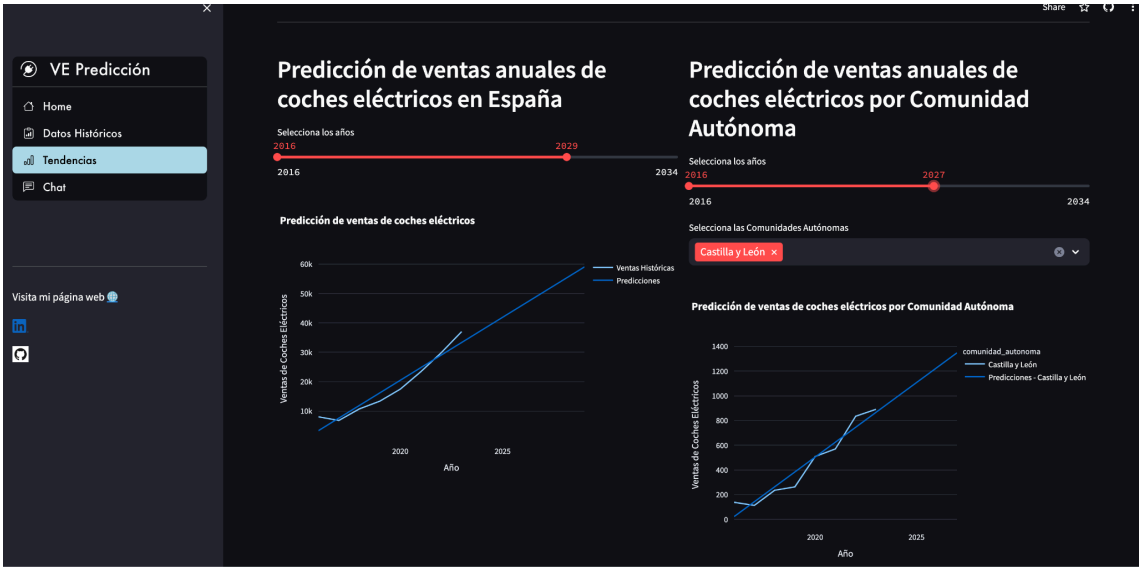
A continuación, se presentan las diferentes pantallas de la página web:

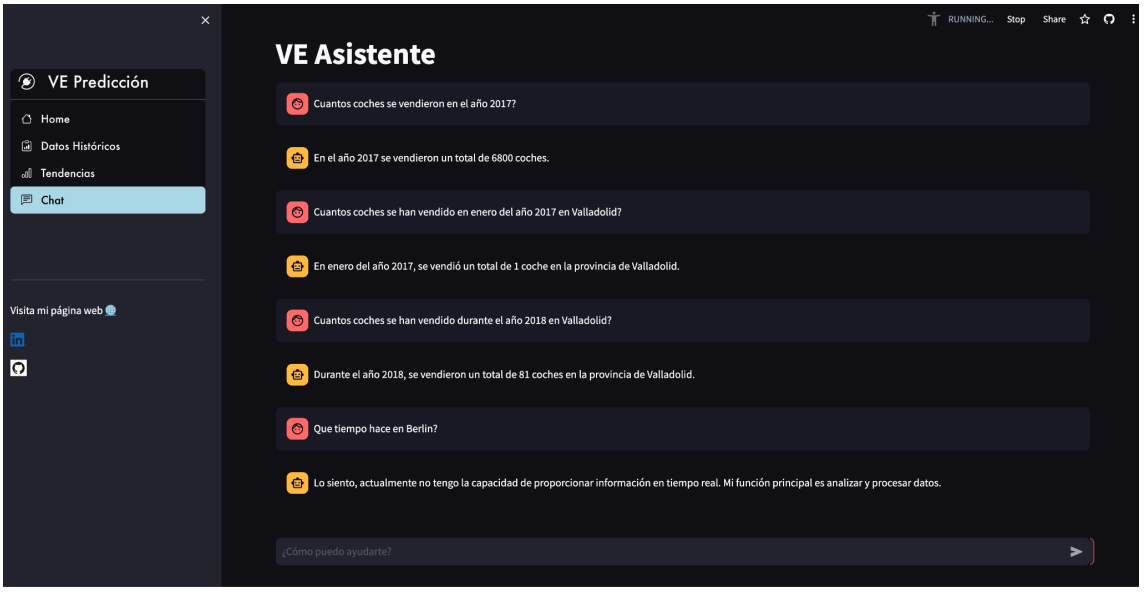



VE Predicción
Home
Datos Históricos
Tendencias
Chat
Visita mi página web
Ventas Anuales por Comunidad Autónoma
Selecciona el año
2019
Selecciona el año
2019
Distribución de ventas de coches eléctricos por Comunidad Autónoma
Catalunya 22.2%
Madrid 43.1%
Andalucía 8.43%
unitat Valenciana 5.02%
Canarias 3.64%
Balears (Illes) 3.16%
País Vasco 3.08%
Castilla-La Mancha 2.16%
Melilla 0.0124%
Ceuta 0.0124%
Rioja 0.211%
Madrid
Catalunya
Andalucía
Comunitat Valenciana
Canarias
Balears (Illes)
País Vasco
Castilla-La Mancha
Galicia
Castilla y León
Aragón
Asturias
Murcia
Navarra
Venta anual de coches por Comunidad Autónoma
Total de Coches
3000
2000
1000
0
Andalucía
Aragón
Asturias
Balears (Illes)
Canarias
Cantabria
Castilla y León
Castilla-La Mancha
Catalunya
Ceuta
Comunitat Valenciana
Extremadura
Galicia
Madrid
Melilla
Murcia
Navarra
País Vasco
Rioja
Comunidad Autónoma
Made in with by @CintiaGarciaGarces
Manage app


VE Predicción
Home
Datos Históricos
Tendencias
Chat
Visita mi página web
Predicciones Ventas Coches Eléctricos
Predicción ventas de coches eléctricos
Selecciona los años futuros
1
9
11
Predicción de ventas
15k
10k
5k
0
Ventas
2016
2018
2020
2022
2024
2026
2028
2030
2032
Fecha
Ventas Históricas
Predicciones
Límite inferior
Límite superior
Made in with by @CintiaGarciaGarces
Manage app


VE Predicción
Home
Datos Históricos
Tendencias
Chat
Visita mi página web
Predicción ventas de coches eléctricos por Comunidad Autónoma
Selecciona una Comunidad Autónoma
Madrid
Selecciona los años futuros para Madrid
9
1
11
Predicción de ventas para Madrid
Ventas Históricas
Predicciones
Límite inferior
Límite superior
Ventas
Fecha


VE Predicción
Home
Datos Históricos
Tendencias
Chat
Visita mi página web
Predicción de ventas anuales de coches eléctricos en España
Selecciona los años
2016
2029
2016
2034
Predicción de ventas de coches eléctricos
Ventas Históricas
Predicciones
Ventas de Coches Eléctricos
Año
Predicción de ventas anuales de coches eléctricos por Comunidad Autónoma
Selecciona los años
2016
2027
2016
2034
Selecciona las Comunidades Autónomas
Castilla y León
Predicción de ventas de coches eléctricos por Comunidad Autónoma
comunidad_autonoma
Castilla y León
Predicciones - Castilla y León
Ventas de Coches Eléctricos
Año

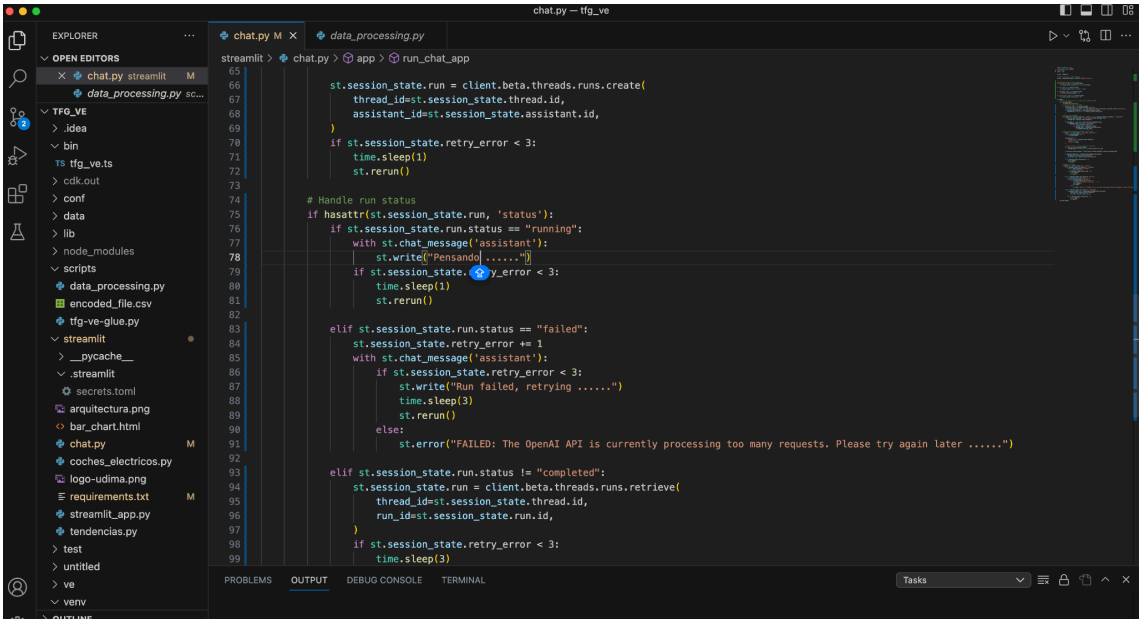

VE Predicción
Home
Datos Históricos
Tendencias
Chat
Visita mi página web
RUNNING...
Stop
Share
VE Asistente
Cuantos coches se vendieron en el año 2017?
En el año 2017 se vendieron un total de 6800 coches.
Cuantos coches se han vendido en enero del año 2017 en Valladolid?
En enero del año 2017, se vendió un total de 1 coche en la provincia de Valladolid.
Cuantos coches se han vendido durante el año 2018 en Valladolid?
Durante el año 2018, se vendieron un total de 81 coches en la provincia de Valladolid.
Que tiempo hace en Berlin?
Lo siento, actualmente no tengo la capacidad de proporcionar información en tiempo real. Mi función principal es analizar y procesar datos.
¿Cómo puedo ayudarte?

Anexo A5. Código

El código puede encontrase en github en el siguiente enlace: https://github.com/cintiagarcia/tfg-ve

La página web se encuentra: web-app

Estructura del código



Scripts:

data_processing.py:

```
profile_name = 'default'
region_name = 'eu-central-1'

session = boto3.Session(profile_name=profile_name, region_name=region_name)
```

```
def encode_csv_to_unicode(file_path):

    # Leer el archivo CSV en un DataFrame de pandas
    df = pd.read_csv(file_path, encoding='latin-1')

    # Agregar nueva columna al archivo CSV
    df['Year'] = year
    df['month'] = str(month).zfill(2)

    # Guardar el DataFrame codificado en un nuevo archivo CSV
    encoded_file_path = 'encoded_file.csv'
    df.to_csv(encoded_file_path, index=False)

    return encoded_file_path

def upload_file_to_s3(bucket_name, file_path, s3_key, kms_key_arn):
    s3 = boto3.client('s3')

    # Subir el archivo al bucket de AWS S3
      s3.upload_file(file_path, bucket_name, s3_key, ExtraArgs={'SSEKMSKeyId':
kms_key_arn, 'ServerSideEncryption': 'aws:kms'})

    print(f"File uploaded to S3 bucket: s3://{bucket_name}/{s3_key}")

bucket_name = 'raw-data-ve-eu-central-1'
s3_key_prefix = 'data/'
kms_key_arn                                                              =
'arn:aws:kms:eu-central-1:766973746059:key/4ae649c9-0b3f-4080-8344-2b0c7696e44
a'

# Recorrer la estructura de directorios
for year in range(2016, 2024):
    for month in range(1, 13):
        # Generar la ruta del archivo basada en la estructura de directorios
                                                              file_path        =
f"/Users/I559673/Documents/Informatica/tfg_ve/data/{year}/{str(month).zfill(2)
}/InformePredefinido_Parque{year}{str(month).zfill(2)}.csv"

        # Comprobar si el archivo existe
        if os.path.exists(file_path):
```

```
            # Codificar el archivo CSV a Unicode
            encoded_file_path = encode_csv_to_unicode(file_path)

            # Generar la clave de S3 basada en la estructura de directorios
                                                                    s3_key      =
f"{s3_key_prefix}{year}/{str(month).zfill(2)}/informe_VE_{year}_{str(month).zf
ill(2)}.csv"

            # Subir el archivo codificado a AWS S3
                    upload_file_to_s3(bucket_name,  encoded_file_path,  s3_key,
kms_key_arn)
        else:
            print(f"File does not exist: {file_path}")
```

tgg-ve-glue.py

```
def   sparkSqlQuery(glueContext,   query,   mapping,   transformation_ctx)   ->
DynamicFrame:
    for alias, frame in mapping.items():
        frame.toDF().createOrReplaceTempView(alias)
    result = spark.sql(query)
    return DynamicFrame.fromDF(result, glueContext, transformation_ctx)


args = getResolvedOptions(sys.argv, ["JOB_NAME"])
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

# Script generated for node Amazon S3
AmazonS3_node1700937060089 = glueContext.create_dynamic_frame.from_options(
    format_options={
        "quoteChar": '"',
        "withHeader": True,
        "separator": ";",
```

```
        "optimizePerformance": False,
    },
    connection_type="s3",
    format="csv",
    connection_options={
        "paths": ["s3://raw-data-ve-eu-central-1/data/"],
        "recurse": True,
    },
    transformation_ctx="AmazonS3_node1700937060089",
)

# Script generated for node Change Schema
ChangeSchema_node1700937094143 = ApplyMapping.apply(
    frame=AmazonS3_node1700937060089,
    mappings=[
        (
            "comunidad autï¿½noma/provincia residencia",
            "string",
            "comunidad_autonoma/provincia",
            "string",
        ),
        ("tipo de vehï¿½culo", "string", "vehiculo", "string"),
        ("carburante", "string", "carburante", "string"),
        ("total,year,month", "string", "total/ano/mes", "string"),
    ],
    transformation_ctx="ChangeSchema_node1700937094143",
)

# Script generated for node SQL Query
SqlQuery466 = """
SELECT
    split(`comunidad_autonoma/provincia`, '/')[0] AS comunidad_autonoma,
    split(`comunidad_autonoma/provincia`, '/')[1] AS provincia,
    vehiculo,
    carburante,
    split(`total/ano/mes`, ',')[0] AS total,
    split(`total/ano/mes`, ',')[1] AS ano,
    split(`total/ano/mes`, ',')[2] AS mes
FROM veData
WHERE carburante != 'TOTAL' and vehiculo is not null;
```

```
"""
SQLQuery_node1700937621405 = sparkSqlQuery(
    glueContext,
    query=SqlQuery466,
    mapping={"veData": ChangeSchema_node1700937094143},
    transformation_ctx="SQLQuery_node1700937621405",
)

# Script generated for node SQL Query
SqlQuery467 = """
SELECT
    CASE
        WHEN comunidad_autonoma = 'Andalucï¿½a' THEN 'Andalucía'
        WHEN comunidad_autonoma = 'Aragï¿½n' THEN 'Aragón'
        WHEN comunidad_autonoma = 'Castilla y Leï¿½n' THEN 'Castilla y León'
        WHEN comunidad_autonoma = 'Cataluï¿½a' THEN 'Catalunya'
        WHEN comunidad_autonoma = 'Murcia (Regiï¿½n de)' THEN 'Murcia'
        WHEN comunidad_autonoma = 'Paï¿½s Vasco' THEN 'País Vasco'
        WHEN comunidad_autonoma = 'Asturias (Principado de)' THEN 'Asturias'
        WHEN comunidad_autonoma = 'Madrid (Comunidad de)' THEN 'Madrid'
        WHEN comunidad_autonoma = 'Madrid (Comunidad de)' THEN 'Madrid'
        WHEN comunidad_autonoma = 'Navarra (Comunidad Foral de)' THEN 'Navarra'
        WHEN comunidad_autonoma = 'Rioja (La)' THEN 'Rioja'
        ELSE comunidad_autonoma
    END AS comunidad_autonoma,
    CASE
        WHEN provincia = 'Almerï¿½a' THEN 'Almería'
        WHEN provincia = 'Cï¿½diz' THEN 'Cádiz'
        WHEN provincia = 'Cï¿½rdoba' THEN 'Córdoba'
        WHEN provincia = 'Jaï¿½n' THEN 'Jaén'
        WHEN provincia = 'Mï¿½laga' THEN 'Málaga'
        WHEN provincia = 'ï¿½vila' THEN 'Ávila'
        WHEN provincia = 'Leï¿½n' THEN 'León'
        WHEN provincia = 'Castellï¿½n' THEN 'Castellón'
        WHEN provincia = 'Cï¿½ceres' THEN 'Cáceres'
        WHEN provincia = 'Coruï¿½a (A)' THEN 'La Coruña'
        WHEN provincia = 'Rioja (La)' THEN 'Rioja'
        ELSE provincia
    END AS provincia,
    vehiculo,
```

```
    CASE
        WHEN carburante = 'Eli¿½ctrico' THEN 'eléctrico'
        ELSE carburante
    END AS carburante,
    total,
    ano,
    mes
FROM veData;
"""
SQLQuery_node1700938286079 = sparkSqlQuery(
    glueContext,
    query=SqlQuery467,
    mapping={"veData": SQLQuery_node1700937621405},
    transformation_ctx="SQLQuery_node1700938286079",
)

# Script generated for node Change Schema
ChangeSchema_node1700938413030 = ApplyMapping.apply(
    frame=SQLQuery_node1700938286079,
    mappings=[
        ("comunidad_autonoma", "string", "comunidad_autonoma", "string"),
        ("provincia", "string", "provincia", "string"),
        ("vehiculo", "string", "vehiculo", "string"),
        ("carburante", "string", "carburante", "string"),
        ("total", "string", "total", "string"),
        ("ano", "string", "año", "string"),
        ("mes", "string", "mes", "string"),
    ],
    transformation_ctx="ChangeSchema_node1700938413030",
)

# Script generated for node Clean data  - ve
Cleandatave_node1701336191329 = glueContext.write_dynamic_frame.from_options(
    frame=ChangeSchema_node1700938413030,
    connection_type="s3",
    format="csv",
            connection_options={"path":     "s3://clean-data-ve-eu-central-1",
"partitionKeys": []},
    transformation_ctx="Cleandatave_node1701336191329",
)
```

```
job.commit()
```

Streamlit:

streamlit_app.py

```
st.set_page_config(page_title='VE Price Prediction App', layout='wide')

def image(url, width=None, height=None):
       return  f'<img  src="{url}"  {"width="+str(width)  if  width  else  ""}
{"height="+str(height) if height else ""}>'

def link(url, text):
   return f'<a href="{url}">{text}</a>'


footer = """
    <div  style="padding:  10px;  position:  fixed;  bottom:  0;  width:  100%;
text-align: center;">
       Made in {0} with ❤️ by {1}
   </div>

""".format(image('https://avatars3.githubusercontent.com/u/45109972?s=400&v=4'
, width=25, height=25),
           link("https://www.cintiagarciagarces.com/", "@CintiaGarciaGarces"),
       )


def do_presentation():
        st.markdown("<style>h1 { font-family: 'Arial', sans-serif; }</style>",
unsafe_allow_html=True)

       st.markdown('### Bienvenidos al Trabajo Fin de Grado')
```

```
        st.write('Estás visitando el sitio web dedicado a mi trabajo fin de grado. Aquí encontrarás información detallada sobre mi investigación, los objetivos alcanzados, y los resultados obtenidos a lo largo de este proyecto académico.')

       st.write('Este proyecto de fin de grado se basa en el análisis de datos públicos proporcionados por la DGT y el gobierno de España acerca de la actualidad de los vehículos eléctricos en el país. Estos datos han sido cuidadosamente procesados para ofrecer una perspectiva futura del mercado en cuestión')

          st.write('Este trabajo lo estoy realizando para la Universidad a Distancia de Madrid, donde estoy llevando a cabo mi proyecto de fin de grado.')

             st.write('Agradezco tu interés y espero que encuentres esta presentación informativa y esclarecedora. Si tienes alguna pregunta o comentario, no dudes en ponerte en contacto conmigo.')

          st.write('Gracias por visitar mi trabajo fin de grado. ¡Espero que disfrutes explorando los detalles de mi investigación!')

       st.markdown('___')

          st.write('Aqui puede encontrar la arquitectura de alto nivel de la aplicacion VE Predicción')
       arquitectura_path = "streamlit/arquitectura.png"
       st.image(arquitectura_path, caption='Arquitectura alto nivel')

       st.markdown('___')

       logo_path = "streamlit/logo-udima.png"
            st.image(logo_path, caption='Universidad a Distancia de Madrid', width=200)


def do_coches_electricos():
   coches_electricos.app()
```

```
def do_tendencias():
    tendencias.app()

def do_chat():
    chat.app()

def display_linkedin_icon():
                                                    linkedin_icon                    =
"https://content.linkedin.com/content/dam/me/business/en-us/amp/brand-site/v2/
bg/LI-Bug.svg.original.svg"
    linkedin_url = "https://www.linkedin.com/in/cintia-garcia-garces/"

          image_html    =    f'<a    href="{linkedin_url}"    target="_blank"><img
src="{linkedin_icon}" alt="Imagen" style="width: 24px;"></a>'
    st.markdown(image_html, unsafe_allow_html=True)

def display_github_icon():
                                                      github_icon                    =
"https://github.githubassets.com/assets/GitHub-Mark-ea2971cee799.png"
    github_url = "https://github.com/cintiagarcia"

           image_html    =    f'<a    href="{github_url}"    target="_blank"><img
src="{github_icon}" alt="Imagen" style="width: 24px;"></a>'
    st.markdown(image_html, unsafe_allow_html=True)

def display_my_website_link():
    web_url = "https://www.cintiagarciagarces.com/"

    st.sidebar.write(f"Visita mi página web [🌐]({web_url})")

styles = {
       "container":  {"margin":  "0px  !important",  "padding":  "0!important",
"align-items": "stretch", "font-family": "Futura, Sans-serif"},
    "icon": {"font-family": "Futura, sans-serif"},
     "nav-link": {"text-align": "left", "margin":"0px", "font-family": "Futura,
sans-serif"},
       "nav-link-selected":  {"background-color":  "lightblue",  "font-weight":
"normal", "color": "black", "font-family": "Futura, sans-serif"},
}
```

```
menu = {
    'title': 'VE Predicción',
    'items': {
        'Home' : {
            'action': None, 'item_icon': 'house', 'submenu': {
                'title': None,
                'items': {
                                'TFG' : {'action': do_presentation, 'item_icon':
'mortarboard', 'submenu': None},
                            'Datos Históricos' : {'action': do_coches_electricos,
'item_icon': 'clipboard-data', 'submenu': None},
                            'Tendencias' : {'action': do_tendencias, 'item_icon':
'bar-chart', 'submenu': None},
                        'Chat' : {'action': do_chat, 'item_icon': 'chat-left-text',
'submenu': None},
                },
                'menu_icon': None,
                'default_index': 0,
                'with_view_panel': 'main',
                'orientation': 'horizontal',
                'styles': styles
            }
        },
        'Datos Históricos' : {
            'action': None, 'item_icon': 'clipboard-data', 'submenu': {
                'title': None,
                'items': {
                                    'Ventas Coches Eléctricos' : {'action':
do_coches_electricos, 'item_icon': 'ev-front', 'submenu': None},
                },
                'menu_icon': None,
                'default_index': 0,
                'with_view_panel': 'main',
                'orientation': 'horizontal',
                'styles': styles
            }
        },
        'Tendencias' : {
            'action': None, 'item_icon': 'bar-chart', 'submenu': {
                'title': None,
```

```
                'items': {
                            'Predicciones Ventas Coches Eléctricos' : {'action':
do_tendencias, 'item_icon': 'graph-up', 'submenu': None},
                },
                'menu_icon': None,
                'default_index': 0,
                'with_view_panel': 'main',
                'orientation': 'horizontal',
                'styles': styles
            }
        },
        'Chat' : {
            'action': None, 'item_icon': 'chat-left-text', 'submenu': {
                'title': None,
                'items': {
                            'Chat' : {'action': do_chat, 'item_icon': 'wechat',
'submenu': None},
                },
                'menu_icon': None,
                'default_index': 0,
                'with_view_panel': 'main',
                'orientation': 'horizontal',
                'styles': styles
            }
        }
    },

    'menu_icon': 'plugin',
    'default_index': 0,
    'with_view_panel': 'sidebar',
    'orientation': 'vertical',
    'styles': styles
}

def show_menu(menu):
    def _get_options(menu):
        options = list(menu['items'].keys())
        return options

    def _get_icons(menu):
```

```
        icons = [v['item_icon'] for _k, v in menu['items'].items()]
        return icons

    kwargs = {
        'menu_title': menu['title'] ,
        'options': _get_options(menu),
        'icons': _get_icons(menu),
        'menu_icon': menu['menu_icon'],
        'default_index': menu['default_index'],
        'orientation': menu['orientation'],
        'styles': menu['styles']
    }

    with_view_panel = menu['with_view_panel']
    if with_view_panel == 'sidebar':
        with st.sidebar:
            menu_selection = option_menu(**kwargs)

            st.markdown(' ' * 5)
            st.markdown('___')
            display_my_website_link()

            display_linkedin_icon()
            st.empty()
            display_github_icon()

    elif with_view_panel == 'main':
        menu_selection = option_menu(**kwargs)
    else:
         raise ValueError(f"Unknown view panel value: {with_view_panel}. Must be
'sidebar' or 'main'.")

    if menu['items'][menu_selection]['submenu']:
        show_menu(menu['items'][menu_selection]['submenu'])

    if menu['items'][menu_selection]['action']:
        menu['items'][menu_selection]['action']()

    st.markdown(footer, unsafe_allow_html=True)
```

```
show_menu(menu)
```

coches_electricos.py

```
def app():
    # List all objects in the bucket
    bucket_name = "clean-data-ve-eu-central-1"

    @functools.lru_cache(maxsize=None)
    def read_data_from_s3(bucket_name, conn):
        # Crear un objeto de sistema de archivos S3
        fs = s3fs.S3FileSystem()
        file_paths = fs.glob(f"{bucket_name}/*")


        # Leer los objetos y procesar los datos
        data_frames = []
        for file_path in file_paths:
            with fs.open(file_path, "rb") as file:
                # Read the object content, a csv file
                df = pd.read_csv(file)
                data_frames.append(df)

        # Concatenar todos los marcos de datos
        df = pd.concat(data_frames)

        return df

    @st.cache_data
    def generate_yearly_sales_chart(df):
```

```
        # Calcular el sumatorio de la columna "diferencia_mes_anterior" por año
        grouped_data = df.groupby('año')['total_coches'].sum().reset_index()

        barra = px.bar(grouped_data, x='año', y='total_coches')
        linea = px.line(grouped_data, x='año', y='total_coches')

        fig = go.Figure()

                    fig.add_trace(go.Bar(x=barra.data[0].x,   y=barra.data[0].y,
name="Gráfico Barra"))
                fig.add_trace(go.Scatter(x=linea.data[0].x,  y=linea.data[0].y,
name="Gráfico Línea"))

        fig.update_layout(title='Venta anual de coches eléctricos en España')

        return fig

    @st.cache_data
    def show_sales_kpi(df):
        total_sales = df.groupby('año')['total_coches'].sum().reset_index()

                fig  =  px.pie(total_sales,  values='total_coches',  names='año',
title='Distribución de ventas de coches eléctricos por año en España')
                    fig.update_traces(textinfo='percent+label',   pull=[0.1]   *
len(total_sales))

        return fig

    @st.cache_data(experimental_allow_widgets=True)
    def generate_sales_by_region_chart(df):
        unique_years = df["año"].unique()

              selected_year  =  st.selectbox("Selecciona  el  año",  unique_years,
key="sales_by_region_year")

        filter = df[df["año"] == selected_year]

                                                accumulated_per_region       =
filter.groupby('comunidad_autonoma')['total_coches'].sum().reset_index()
```

```
            fig = px.bar(accumulated_per_region, x='comunidad_autonoma',
y='total_coches', title='Venta anual de coches por Comunidad Autónoma')
        fig.update_xaxes(title='Comunidad Autónoma')
        fig.update_yaxes(title='Total de Coches')

        return fig

    @st.cache_data(experimental_allow_widgets=True)
    def show_sales_by_region_kpi(df):
        unique_years = df["año"].unique()

            selected_year = st.selectbox("Selecciona el año", unique_years,
key="sales_by_region_year_kpi")

        filtrado = df[df["año"] == selected_year]

                                                accumulated_per_region      =
filtrado.groupby('comunidad_autonoma')['total_coches'].sum().reset_index()

        fig = px.pie(
            accumulated_per_region,
            values='total_coches',
            names='comunidad_autonoma',
              title='Distribución de ventas de coches eléctricos por Comunidad
Autónoma'
        )
                    fig.update_traces(textinfo='percent+label', pull=[0.1] *
len(accumulated_per_region))

        return fig

    @st.cache_data
    def show_kpis(df):
        total_sales = int(df['total_coches'].sum())
        total_sales_by_year = df.groupby('año')['total_coches'].sum()

        annual_variation = total_sales_by_year.pct_change() * 100
         annual_variation = annual_variation.replace([np.inf, -np.inf, np.nan],
0)
```

```
        with st.container():
            col1, col2 = st.columns((2, 6))
            with col1:
                st.markdown('''
                <div style="border: 2px solid #45a7c8; padding: 10px">
                    <h3 style="text-align: center">Total de ventas</h3>
                                        <p  style="text-align:  center;  font-size:
24px"><strong>{}</strong> <i class="bi bi-ev-front"></i></p>
                    <br>
                </div>
                '''.format('{:,}'.format(total_sales)), unsafe_allow_html=True)

            with col2:
                st.markdown('''
                <div style="border: 2px solid #45a7c8; padding: 10px">
                         <h3 style="text-align: center">Porcentaje de Variación
Anual</h3>
                    <div style="overflow-x: auto;">
                        <table style="margin-left: auto; margin-right: auto;">
                            <thead>
                                <tr>
                                    <th></th>
                                    {}
                                </tr>
                            </thead>
                            <tbody>
                                <tr>
                                        <td style="text-align: center">Porcentaje
de Variación</td>
                                    {}
                                </tr>
                            </tbody>
                        </table>
                    </div>
                </div>
                '''.format(
                                    ''.join('<th>{}</th>'.format(año)  for  año  in
total_sales_by_year.index),
                                                ''.join('<td  style="text-align:
center">{:.2f}%</td>'.format(variación) for variación in annual_variation),
```

```
            ), unsafe_allow_html=True)

    @st.cache_data(experimental_allow_widgets=True)
    def generate_yearly_sales_cum_chart(df):
            df['fecha'] = pd.to_datetime(df['año'].astype(str) + '-' +
df['mes'].astype(str))

        df = df.sort_values('fecha')

        df['acumulado'] = df['total_coches'].cumsum()

        fig = go.Figure()
                fig.add_trace(go.Scatter(x=df['fecha'], y=df['acumulado'],
mode='lines', name='Acumulado de Ventas', line_shape='spline'))

        st.header("Número Total de Coches Eléctricos")

         fig.update_layout(title='Número de coches eléctricos en circulación en
España por Mes',
                        xaxis_title='Fecha',
                        yaxis_title='Número de Coches Eléctricos')

        return fig

    conn = st.connection('s3', type=FilesConnection)

    df = read_data_from_s3(bucket_name, conn)

    st.markdown('---')

    st.write("")

    show_kpis(df)

    st.write("")

    st.markdown('---')

        st.markdown('<h1 style="text-align: center;">Ventas Anuales</h1>',
unsafe_allow_html=True)
```

```
    col1, col2 = st.columns((5,5))
    with col1:
        total_sales_kpi = show_sales_kpi(df)
        st.plotly_chart(total_sales_kpi, use_container_width=True)
    with col2:
        yearly_sales_chart = generate_yearly_sales_chart(df)
        st.plotly_chart(yearly_sales_chart, use_container_width=True)

    st.markdown('---')

     st.markdown('<h1 style="text-align: center;">Ventas Anuales por Comunidad
Autónoma</h1>', unsafe_allow_html=True)

    col1, col2 = st.columns((5,5))
    with col1:
        yearly_sales_kpi = show_sales_by_region_kpi(df)
        st.plotly_chart(yearly_sales_kpi, use_container_width=True)
    with col2:
        sales_by_region_chart = generate_sales_by_region_chart(df)
        st.plotly_chart(sales_by_region_chart, use_container_width=True)

    st.write("")

    st.markdown('---')

    sales_by_region_chart = generate_yearly_sales_cum_chart(df)
    st.plotly_chart(sales_by_region_chart, use_container_width=True)

    st.markdown('---')
```

tendencias.py

```

```

```
def app():
    # Cargar los datos desde el S3 bucket
    bucket_name = "clean-data-ve-eu-central-1"

    @functools.lru_cache(maxsize=None)
    def read_data_from_s3(bucket_name, conn):
        # Create an S3 file system object
        fs = s3fs.S3FileSystem()

        file_paths = fs.glob(f"{bucket_name}/*")

        # Read the objects and process the data
        data_frames = []
        for file_path in file_paths:
            with fs.open(file_path, "rb") as file:
                # Read the object content, a csv file
                df = pd.read_csv(file)
                data_frames.append(df)

        # Concatenate all data frames
        df = pd.concat(data_frames)

        return df

    # Conexión con S3 bucket
    conn = st.connection('s3', type=FilesConnection)
    # Cargar los datos
    df = read_data_from_s3(bucket_name, conn)

    @st.cache_data(experimental_allow_widgets=True)
    def predecir_ventas_con_prophet(df, selected_years):
                                df_grouped    =    df.groupby(['año',    'mes'],
as_index=False)['total_coches'].sum().reset_index()
        df_grouped['ds'] = pd.to_datetime(df_grouped['año'].astype(str) + '-' +
df_grouped['mes'].astype(str) + '-01')
        df_grouped.rename(columns={'total_coches': 'y'}, inplace=True)

        df_prophet = df_grouped[['ds', 'y']]

        model = Prophet()
```

```
        model.fit(df_prophet)

                    future_dates   =   pd.date_range(start=df_prophet['ds'].max(),
periods=selected_years * 12, freq='M')
        future_df = pd.DataFrame({'ds': future_dates})

        forecast = model.predict(future_df)

        fig = go.Figure()

        # Datos históricos
                fig.add_trace(go.Scatter(x=df_prophet['ds'],  y=df_prophet['y'],
mode='lines', name='Ventas Históricas'))

        # Predicciones
                 fig.add_trace(go.Scatter(x=forecast['ds'],  y=forecast['yhat'],
mode='lines', name='Predicciones'))
            fig.add_trace(go.Scatter(x=forecast['ds'], y=forecast['yhat_lower'],
name='Límite inferior', line=dict(color='red')))
            fig.add_trace(go.Scatter(x=forecast['ds'], y=forecast['yhat_upper'],
name='Límite superior', line=dict(color='red')))

        # Configuración del diseño del gráfico
        fig.update_layout(
            title='Predicción de ventas',
            xaxis=dict(title='Fecha'),
            yaxis=dict(title='Ventas')
        )

        return fig, forecast

    st.markdown('---')
    st.title('Predicción ventas de coches eléctricos')
    selected_years = st.slider('Selecciona los años futuros', 1, 11, 5)
    fig, df_forecast = predecir_ventas_con_prophet(df, selected_years)

    st.plotly_chart(fig, use_container_width=True)

    df_forecast = df_forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']]
```

```
        df_forecast.rename(columns={'ds':   'Fecha',   'yhat':   'Predicciones',
'yhat_lower':    'Límite    inferior',    'yhat_upper':    'Límite    superior'},
inplace=True)
   st.subheader('Predicciones de Ventas')
   st.dataframe(df_forecast)
   st.markdown('---')

   @st.cache_data(experimental_allow_widgets=True)
         def   predecir_ventas_con_prophet_por_comunidad(df,   selected_years,
selected_comunidad):
       df_filtered = df[df['comunidad_autonoma'] == selected_comunidad]

                       df_grouped    =    df_filtered.groupby(['año',    'mes'],
as_index=False)['total_coches'].sum().reset_index()
        df_grouped['ds'] = pd.to_datetime(df_grouped['año'].astype(str) + '-' +
df_grouped['mes'].astype(str) + '-01')
       df_grouped.rename(columns={'total_coches': 'y'}, inplace=True)

       df_prophet = df_grouped[['ds', 'y']]

       model = Prophet()

       model.fit(df_prophet)

                  future_dates   =   pd.date_range(start=df_prophet['ds'].max(),
periods=selected_years * 12, freq='M')
       future_df = pd.DataFrame({'ds': future_dates})

       forecast = model.predict(future_df)

       fig = go.Figure()

              fig.add_trace(go.Scatter(x=df_prophet['ds'],  y=df_prophet['y'],
mode='lines', name='Ventas Históricas'))

               fig.add_trace(go.Scatter(x=forecast['ds'],  y=forecast['yhat'],
mode='lines', name='Predicciones'))
          fig.add_trace(go.Scatter(x=forecast['ds'], y=forecast['yhat_lower'],
name='Límite inferior', line=dict(color='red')))
```

```
            fig.add_trace(go.Scatter(x=forecast['ds'], y=forecast['yhat_upper'],
name='Límite superior', line=dict(color='red')))

        fig.update_layout(
            title=f'Predicción de ventas para {selected_comunidad}',
            xaxis=dict(title='Fecha'),
            yaxis=dict(title='Ventas')
        )

        return fig, forecast

    st.title('Predicción ventas de coches eléctricos por Comunidad Autónoma')

    comunidades_autonomas = df['comunidad_autonoma'].unique()
      selected_comunidad = st.selectbox('Selecciona una Comunidad Autónoma',
comunidades_autonomas)

         selected_years = st.slider(f'Selecciona los años futuros para
{selected_comunidad}', 1, 11, 5)

          fig, df_forecast = predecir_ventas_con_prophet_por_comunidad(df,
selected_years, selected_comunidad)
    st.plotly_chart(fig, use_container_width=True)

    df_forecast = df_forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']]
         df_forecast.rename(columns={'ds': 'Fecha', 'yhat': 'Predicciones',
'yhat_lower': 'Límite inferior', 'yhat_upper': 'Límite superior'},
inplace=True)
    st.subheader('Predicciones de Ventas')
    st.dataframe(df_forecast)

    @st.cache_data(experimental_allow_widgets=True)
    def predecir_ventas_coches_electricos(df, selected_years):
        df_anual = df.groupby('año')['total_coches'].sum().reset_index()

        # Dividir los datos en variables de entrada (X) y variable objetivo (y)
        X = df_anual['año'].values.reshape(-1, 1)
        y = df_anual['total_coches'].values.reshape(-1, 1)

        regression_model = LinearRegression()
```

```
        regression_model.fit(X, y)

            future_dates = np.arange(selected_years[0], selected_years[1] +
1).reshape(-1, 1)

        future_predictions = regression_model.predict(future_dates)

         df_pred = pd.DataFrame({'año': future_dates.flatten(), 'Predicciones':
future_predictions.flatten()})

        # Combinar datos históricos y predicciones
        df_combined = pd.concat([df_anual, df_pred])

        fig = go.Figure()

        fig.add_trace(go.Scatter(x=df_anual['año'], y=df_anual['total_coches'],
mode='lines', name='Ventas Históricas'))

          fig.add_trace(go.Scatter(x=df_pred['año'], y=df_pred['Predicciones'],
mode='lines', name='Predicciones'))

        fig.update_layout(
            title='Predicción de ventas de coches eléctricos',
            xaxis=dict(title='Año'),
            yaxis=dict(title='Ventas de Coches Eléctricos')
        )
        return fig


    @st.cache_data(experimental_allow_widgets=True)
    def predecir_ventas_coches_por_comunidad(df, selected_years_comunidad):
                              df_anual    =    df.groupby(['comunidad_autonoma',
'año'])['total_coches'].sum().reset_index()

        comunidades_autonomas = df_anual['comunidad_autonoma'].unique()

            selected_comunidades = st.multiselect('Selecciona las Comunidades
Autónomas', comunidades_autonomas)
```

```
        if len(selected_comunidades) == 0:
            # Devolver un gráfico vacío cuando no hay selecciones
            return go.Figure()

                                                        df_filtered        =
df_anual[df_anual['comunidad_autonoma'].isin(selected_comunidades)]

        prediction_dfs = []

        # Iterar sobre cada comunidad autónoma seleccionada
        for comunidad_autonoma in selected_comunidades:
                df_comunidad = df_filtered[df_filtered['comunidad_autonoma'] ==
comunidad_autonoma]

            X = df_comunidad['año'].values.reshape(-1, 1)
            y = df_comunidad['total_coches'].values.reshape(-1, 1)

            regression_model = LinearRegression()

            regression_model.fit(X, y)

                        future_dates  =  np.arange(selected_years_comunidad[0],
selected_years_comunidad[1] + 1).reshape(-1, 1)

            future_predictions = regression_model.predict(future_dates)

                      df_pred  =  pd.DataFrame({'año':  future_dates.flatten(),
'Predicciones': future_predictions.flatten()})

            df_pred['Predicciones'] = df_pred['Predicciones'].astype(int)

            df_pred['comunidad_autonoma'] = comunidad_autonoma

            prediction_dfs.append(df_pred)

               # Concatenar todos los DataFrames de predicciones por comunidad
autónoma
            df_pred_combined = pd.concat(prediction_dfs)
```

```
                                                df_filtered['total_coches']    =
df_filtered['total_coches'].astype(int)

            # Combinar datos históricos y predicciones por comunidad autónoma
            df_combined = pd.concat([df_filtered, df_pred_combined])

                        fig  =  px.line(df_combined,  x='año',  y='total_coches',
color='comunidad_autonoma',  title='Predicción  de  ventas  de  coches  eléctricos
por Comunidad Autónoma')

            for comunidad_autonoma in selected_comunidades:
                                                            df_pred_comunidad    =
df_pred_combined[df_pred_combined['comunidad_autonoma'] == comunidad_autonoma]
                            fig.add_trace(go.Scatter(x=df_pred_comunidad['año'],
y=df_pred_comunidad['Predicciones'],    mode='lines',    name=f'Predicciones    -
{comunidad_autonoma}'))

            fig.update_layout(
                xaxis=dict(title='Año'),
                yaxis=dict(title='Ventas de Coches Eléctricos')
            )
        return fig

    st.markdown('---')

    col1, col2 = st.columns((5,5))

    with col1:
             st.header("Predicción  de  ventas  anuales  de  coches  eléctricos  en
España")
            selected_years  =  st.slider('Selecciona  los  años',  df['año'].min(),
df['año'].max()              +              11,              (df['año'].min(),
df['año'].max()),key='coches_electricos')
                                                    predecir_ventas_coches       =
predecir_ventas_coches_electricos(df,selected_years)
        st.plotly_chart(predecir_ventas_coches, use_container_width=True)
    with col2:
            st.header("Predicción  de  ventas  anuales  de  coches  eléctricos  por
Comunidad Autónoma")
```

```
            selected_years_comunidad = st.slider('Selecciona los años',
df['año'].min(),      df['año'].max()      +      11,      (df['año'].min(),
df['año'].max()),key='coches_electricos_comunidad')
                                        predecir_ventas_coches_comunidad     =
predecir_ventas_coches_por_comunidad(df, selected_years_comunidad)
                            st.plotly_chart(predecir_ventas_coches_comunidad,
use_container_width=True)



   st.markdown('---')
```

chat.py

```
import streamlit as st
from openai import OpenAI
import uuid
import time

client = OpenAI()

# Definir la clave de API de OpenAI
client = OpenAI(api_key=st.secrets["OPENAI_API_KEY"])



# Initialize session state variables
if "session_id" not in st.session_state:
   st.session_state.session_id = str(uuid.uuid4())

if "run" not in st.session_state:
   st.session_state.run = {"status": None}

if "messages" not in st.session_state:
   st.session_state.messages = []

if "retry_error" not in st.session_state:
```

```
    st.session_state.retry_error = 0

def app():
    # Función para generar una respuesta del modelo ChatGPT
    def run_chat_app():
        st.title("VE Asistente")
        # Initialize OpenAI assistant
        if "assistant" not in st.session_state:
            client.api_key = st.secrets["OPENAI_API_KEY"]
                                                st.session_state.assistant     =
client.beta.assistants.retrieve(st.secrets["OPENAI_ASSISTANT"])
            st.session_state.thread = client.beta.threads.create(
                metadata={'session_id': st.session_state.session_id}
            )

        # Display chat messages
                        elif    hasattr(st.session_state.run,    'status')    and
st.session_state.run.status == "completed":
            st.session_state.messages = client.beta.threads.messages.list(
                thread_id=st.session_state.thread.id
            )
            for message in reversed(st.session_state.messages.data):
                if message.role in ["user", "assistant"]:
                    with st.chat_message(message.role):
                        for content_part in message.content:
                            message_text = content_part.text.value
                            st.markdown(message_text)

        # Chat input and message creation with file ID
        if prompt := st.chat_input("¿Cómo puedo ayudarte?"):
            with st.chat_message('user'):
                st.write(prompt)

            message_data = {
                "thread_id": st.session_state.thread.id,
                "role": "user",
                "content": prompt
            }

            # Include file ID in the request if available
```

```
            if "file_id" in st.session_state:
                message_data["file_ids"] = [st.session_state.file_id]

                                                st.session_state.messages    =
client.beta.threads.messages.create(**message_data)

            st.session_state.run = client.beta.threads.runs.create(
                thread_id=st.session_state.thread.id,
                assistant_id=st.session_state.assistant.id,
            )
            if st.session_state.retry_error < 3:
                time.sleep(1)
                st.rerun()

        # Handle run status
        if hasattr(st.session_state.run, 'status'):
            if st.session_state.run.status == "running":
                with st.chat_message('assistant'):
                    st.write("Pensando ......")
                if st.session_state.retry_error < 3:
                    time.sleep(1)
                    st.rerun()

            elif st.session_state.run.status == "failed":
                st.session_state.retry_error += 1
                with st.chat_message('assistant'):
                    if st.session_state.retry_error < 3:
                        st.write("Run failed, retrying ......")
                        time.sleep(3)
                        st.rerun()
                    else:
                                    st.error("FAILED: The OpenAI API is currently
processing too many requests. Please try again later ......")

            elif st.session_state.run.status != "completed":
                st.session_state.run = client.beta.threads.runs.retrieve(
                    thread_id=st.session_state.thread.id,
                    run_id=st.session_state.run.id,
                )
                if st.session_state.retry_error < 3:
```

```
                time.sleep(3)
                st.rerun()
run_chat_app()
```