Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Uso de modelos de lenguaje y Machine Learning para la consulta y predicción de pasajeros en TITSA

Use of Language Models and Machine Learning for Passenger Query and Prediction in TITSA

Adrián Hernández Suárez

D. **Jose Luis Roda García**, con N.I.F. 43.356.123-L profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

D. **Ginés León Rodríguez**, con N.I.F. 78.408.712-X responsable del Departamento de Big Data & Data Science de T.I.T.S.A, como cotutor

C E R T I F I C A N

Que la presente memoria titulada:

"Uso de modelos de lenguaje y Machine Learning para la consulta y predicción de pasajeros en TITSA"

ha sido realizada bajo su dirección por D. **Adrián Hernández Suárez**,

con N.I.F. 46.298.286-E.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 25 de mayo de 2023

Agradecimientos

Me gustaría expresar mi profunda gratitud a mi familia, mi pareja y amigos por el inmenso apoyo que me han brindado a lo largo de mi carrera. Sin su apoyo constante, muchas de las metas y objetivos que he logrado alcanzar no habrían sido posibles.

Asimismo, deseo agradecer a las personas extraordinarias que he conocido durante mi último año en la Cátedra Cajasiete Blockchain, Open Data & Big Data, en especial a José Luis Roda, Isabel Sánchez, Jose Carlos González y Carlos Dominguez por todo el conocimiento que he adquirido y por el camino que hemos y estamos haciendo, gracias por despertar en mí esta pasión por la Inteligencia Artificial y por guiarme en este aprendizaje.

Por otro lado, me gustaría agradecer profundamente a Ginés León por su labor como tutor externo, y por toda la ayuda y buenos consejos que ha aportado, ha sido una parte clave para buscar ese punto de más a cada una de las fases, y no solo como tutor, también agradecerle como persona, por todo su apoyo.

No podría terminar unos agradecimientos sin nombrar a las personas que en una primera instancia hace 4 años eran compañeros de clase y a día de hoy se han convertido en una parte importante de mi vida, como son, Raúl Martín, Acoidán Mesa, Borja Guanche, Diego Rodríguez, Sergio Pitti y Alberto Ríos. A lo largo de la carrera hemos demostrado que uniendo grandes fuerzas en diferentes áreas se pueden crear equipos y grupos de trabajo excepcionales.

Licencia

Resumen

T.I.T.S.A es una empresa pública de Transportes Interurbanos de Tenerife. La empresa transporta diariamente a más de 150.000 usuarios, en sus más de 5.285 viajes. Esta cantidad tan elevada de pasajeros hace que determinadas líneas de transporte se saturen en las horas de mayor tráfico. Además, con la reciente implementación de la gratuidad para determinados abonos de viaje, este número de pasajeros se ha visto incrementado un 45% en los primeros meses de 2023 con respecto al mismo periodo del año anterior.

El presente proyecto tiene como objetivo principal la creación de una herramienta innovadora que combina modelos de lenguaje como GPT (Generative Pre-trained Transformer) y técnicas de Machine Learning. Esta herramienta servirá para dos propósitos fundamentales: facilitar la consulta de datos históricos de pasajeros a través de un sistema conversacional y emplear algoritmos de Machine Learning para predecir la demanda de pasajeros en una línea de transporte determinada.

Palabras clave: T.I.T.S.A, pasajeros, transporte, Python, SQL, Machine Learning, predicción, series temporales, redes neuronales, Big Data, Open Data, GPT

Abstract

T.I.T.S.A is a public company that provides interurban transport services in Tenerife. The company transports over 150,000 users daily, on its more than 5,285 trips. This high volume of passengers causes certain transport lines to become saturated during peak traffic hours. Furthermore, with the recent implementation of free travel for certain travel passes, the number of passengers has increased by 45% in the first few months of 2023 compared to the same period last year.

The present project aims to create a cutting-edge tool that combines language models like GPT (Generative Pre-trained Transformer) and machine learning techniques. This tool will serve two fundamental purposes: to facilitate the retrieval of historical passenger data through a conversational system and to employ machine learning algorithms to predict passenger demand on a specific transportation line.

Keywords: T.I.T.S.A, passengers, transportation, Python, SQL, Machine Learning, prediction, time series, neural networks, Big Data, Open Data, GPT

Índice General

Índice de figuras

Índice de tablas

Capítulo 1 Introducción

En esta introducción se introduce el problema planteado por la empresa pública Transportes Interurbanos de Tenerife S.A. (TITSA) [1] para poder acceder a los datos históricos sobre pasajeros y para intentar predecir el número de pasajeros en un periodo de tiempo determinado. El objetivo es proporcionar una visión general de los aspectos clave del problema, su complejidad y el alcance, con el fin de contextualizar el trabajo de investigación realizado en este Trabajo Final de Grado.

1.1 Antecedentes

En los últimos años, los datos abiertos, también conocidos como Open Data [2], han pasado de ser una simple aspiración a convertirse en una necesidad fundamental en múltiples sectores. TITSA, en su compromiso por mejorar la calidad de su servicio, creó en 2018 un departamento específico dedicado a la explotación de Big Data y Data Science con el objetivo de utilizar la información disponible para mejorar la gestión de sus líneas de transporte. Entre las tareas de este departamento se incluyen la planificación de rutas, la optimización de las mismas y la realización de predicciones globales de pasajeros, entre otros modelos. Muchas de las aplicaciones desarrolladas en TITSA hacen uso de los datos abiertos de diferentes portales de datos abiertos [3].

Recientemente, TITSA ha experimentado un importante aumento en la cantidad de pasajeros, cercano al 50% en comparación con el mismo periodo del año anterior, tras la introducción de un nuevo sistema tarifario para los abonos de transporte. Este incremento ha generado situaciones de saturación en algunas de las líneas de transporte, por lo que resulta necesario contar con un modelo de predicción de pasajeros por línea que permita reforzar aquellos servicios que se prevén más demandados. Este modelo permitiría prever con suficiente antelación los incrementos de pasajeros, lo que permitiría a TITSA ajustar su oferta de transporte para satisfacer la demanda de manera más eficiente.

1.2 Problemática

Los sistemas de información de TITSA realizan toda la gestión de datos de la empresa. Entre estos sistemas podemos citar la tarificación, la planificación, la operativa, la gestión de stock, o el control de repostaje de combustible. TITSA ha desarrollado un conjunto de modelos y cuadros de mando que les permite obtener información tanto en tiempo real como en histórico de los diferentes sistemas antes descritos, además de crear reportes diarios, semanales o mensuales y enviar correos automáticos a los departamentos en cuestión si se ha detectado algún tipo de error en el sistema, ya sea

por los datos o por el propio software.

Un conjunto de datos que resulta muy valioso para la empresa es la cantidad de pasajeros que se transportan en cada una de las líneas de transporte por cada servicio, identificando la cantidad de pasajeros que entran y salen de la guagua en cada una de las paradas del servicio. Con estos datos y con otras fuentes de datos abiertos externos, como la predicción meteorológica, se podría realizar una predicción a corto plazo incluso llegando a tiempo real de la cantidad de pasajeros que habrá en una parada determinada a una hora y en una línea en cuestión. Por otro lado, TITSA tampoco dispone de un servicio de información instantáneo que ofrezca datos determinados a los responsables de la toma de decisiones. Por tanto en este proyecto se propone resolver ambos problemas con el fin de consultar a la base de datos y dar una respuesta elaborada que pueda ofrecer información del número de pasajeros en un periodo concreto de tiempo, ya sea en pasado o en futuro.

1.3 Alcance del proyecto

Teniendo en cuenta la problemática, el objetivo principal del proyecto es crear una herramienta para la consulta de datos históricos a través de un chatbot de Telegram y realizar un modelo de predicción para comprobar la cantidad esperada de pasajeros por parada sobre una línea de transporte.

Para ello, se utilizarán datos históricos de la cantidad de pasajeros en diferentes líneas de transporte y se desarrollará un modelo de predicción aplicando técnicas como series temporales o modelos de Machine Learning, que permitan predecir la cantidad de pasajeros en una parada y hora determinadas. La herramienta permitirá a los usuarios acceder a esta información de manera sencilla y rápida, a través de la plataforma de Telegram, lo que mejorará la experiencia de los usuarios y contribuirá a una mayor eficiencia en el transporte público. El proyecto incluirá la recopilación y procesamiento de datos, el desarrollo del modelo de predicción, la creación del bot de Telegram y la integración de los diferentes componentes para su funcionamiento óptimo.

Queda fuera del alcance:

- El desarrollo de un chatbot en otro medio de mensajería instantánea como Whatsapp.
- La integración del modelo predictivo en el chatbot de Telegram.
- La creación de un modelo de predicción para múltiples líneas simultáneamente.
- La creación de un modelo de predicción en tiempo real, controlando el flujo de datos.

1.4 Fases y desarrollo del proyecto

A continuación, se presenta un diagrama de Gantt [4] donde se recogen todas las fases en las que se ha dividido el proyecto, así como la duración que estas han tenido:

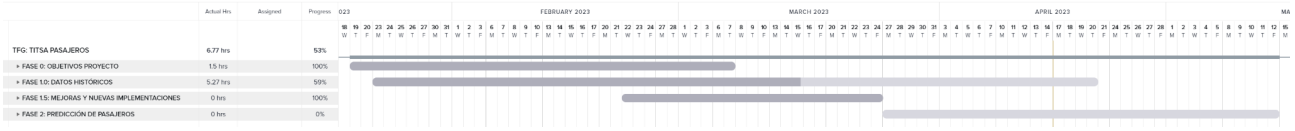

Figura 1.1: Diagrama de Gantt

Desglose de las fases:

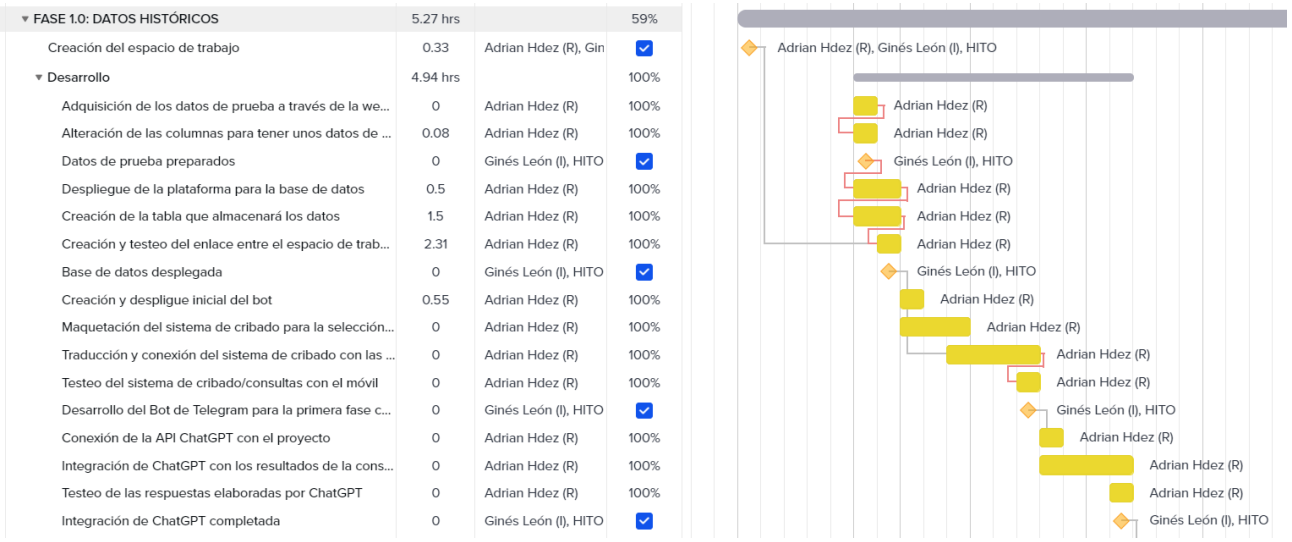

Figura 1.2: Diagrama de Gantt fase I

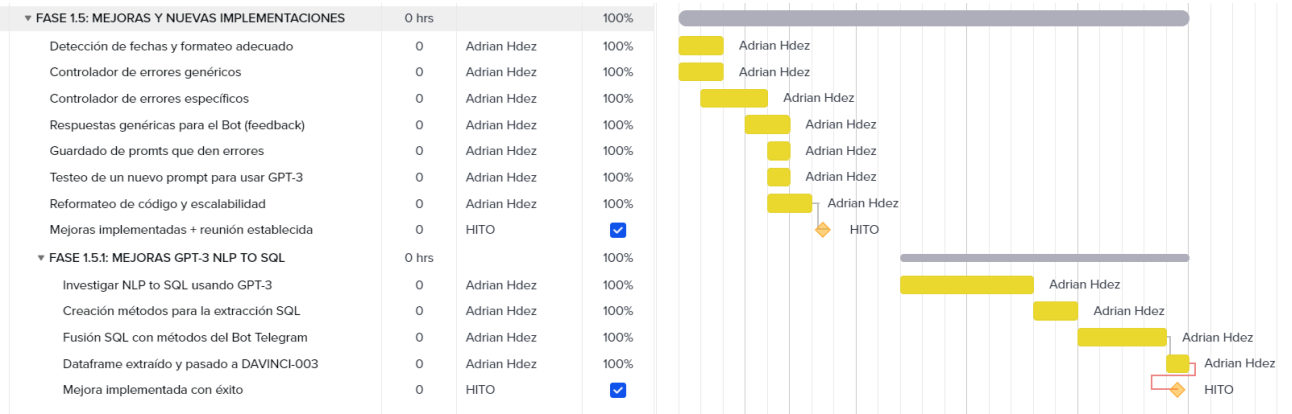

Figura 1.3: Diagrama de Gantt para la mejora fase I

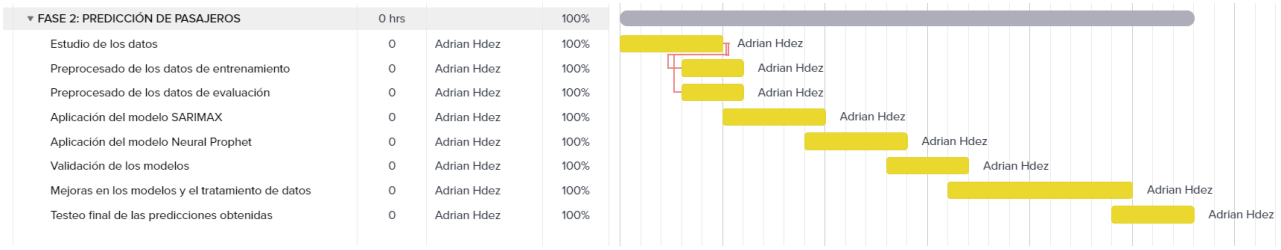

Figura 1.4: Diagrama de Gantt para la fase II

Capítulo 2 Fundamentos teóricos

Para contextualizar el proyecto, se comentarán los principales fundamentos teóricos que subyacen detrás del desarrollo de cada una de las fases. Los fundamentos matemáticos de cada concepto se encontrarán en los enlaces de referencia correspondientes.

2.1 Open Data

El Open Data o "datos abiertos" hace referencia a la práctica de hacer que los datos estén disponibles en línea de forma gratuita y accesible para su reutilización y redistribución sin restricciones de derechos de autor, patentes u otras formas de control de la propiedad intelectual.

El Open Data fomenta la innovación y el desarrollo de nuevas aplicaciones y servicios, lo que puede impulsar la economía y mejorar la calidad de vida de las personas. Al permitir que los desarrolladores utilicen datos públicos en sus aplicaciones, se pueden crear soluciones que ayuden a resolver problemas sociales y ambientales, desde la planificación urbana hasta la gestión de emergencias.

También tiene el potencial de mejorar la eficiencia y la transparencia de las organizaciones. Al publicar datos internos en línea, las organizaciones pueden identificar áreas de mejora y tomar medidas para mejorar su desempeño. Además, al permitir que los ciudadanos accedan a sus datos, las organizaciones pueden mejorar la calidad de sus servicios y mejorar la confianza de sus usuarios.

Llevando el Open Data a nuestro proyecto, el acceso a los datos sobre pasajeros de TITSA puede tener varios beneficios.

1. Mejorar la transparencia y la rendición de cuentas de TITSA, permitiendo a los ciudadanos acceder a la información actualizada sobre la cantidad de pasajeros en las líneas de transporte.
2. Mejorar la planificación de la movilidad urbana y para la toma de decisiones en cuanto a la mejora de los servicios de transporte.
3. Fomentar la innovación y el desarrollo de nuevas aplicaciones y servicios, ya no solo en la propia empresa, sino también en las diferentes administraciones públicas.

2.2 Modelos de predicción de datos

2.2.1 Aprendizaje automático

El aprendizaje automático (*Machine Learning*) [5], es un campo de la inteligencia artificial que se encarga del estudio y desarrollo de algoritmos y modelos capaces de aprender y mejorar su

desempeño a partir de conjuntos de datos. En el contexto de la predicción de datos, el aprendizaje automático se utiliza para entrenar modelos que puedan hacer predicciones precisas y confiables sobre el comportamiento futuro de ciertos fenómenos, como pueden ser ventas, el clima, o en nuestro caso particular, pasajeros de una línea de transporte.

Para lograr una precisión elevada, es necesario entrenar el modelo con datos históricos de calidad y validar su desempeño en un conjunto de datos no utilizado en el entrenamiento, comúnmente conocidos como *validation data*. Además, es importante elegir el modelo adecuado para el problema a resolver y ajustar sus parámetros de manera óptima para evitar ciertos problemas como el *sobreajuste* o el *sub-ajuste* del modelo. Los modelos de Machine Learning más utilizados para la predicción de datos son los modelos de regresión, clasificación, los árboles de decisión, las redes neuronales y los modelos de series temporales, de los cuales hablaremos en profundidad más adelante.

En definitiva, el Machine Learning es una herramienta poderosa para la predicción de datos y su aplicación en diferentes campos puede proporcionar una ventaja competitiva significativa.

2.2.2 Redes Neuronales

Las redes neuronales [6] son una familia de modelos de aprendizaje automático que tratan de esquematizar y modelar la estructura del cerebro humano, para así, poder reproducir sus características computacionales. Es por ello, que su unidad fundamental se denomina neurona la cual realiza ciertas operaciones en base a los datos de entrada, además de tener la capacidad de aprender de la experiencia, haciendo que se vaya auto-ajustando hasta lograr un resultado óptimo.

Estos sistemas de procesamiento de información, paralelos, distribuidos, y adaptativos, se entrenan con un enfoque ascendente, replicando cómo lo hace el cerebro, reconociendo patrones y generalizando a partir de diversos ejemplos.

Bajemos a nivel atómico y comprendamos las redes neuronales desde su inicio. La red neuronal más sencilla, y la que mejor nos puede ayudar a explicar el concepto de red neuronal y su funcionamiento es el *Perceptrón* [7]. El perceptrón plasma a nivel práctico el funcionamiento de una sola neurona, donde en base a varios parámetros de entrada y unos pesos asociados, se realiza una suma ponderada, la neurona se activa y nos genera una salida. En la figura 2.1 se puede observar una representación de un Perceptrón.

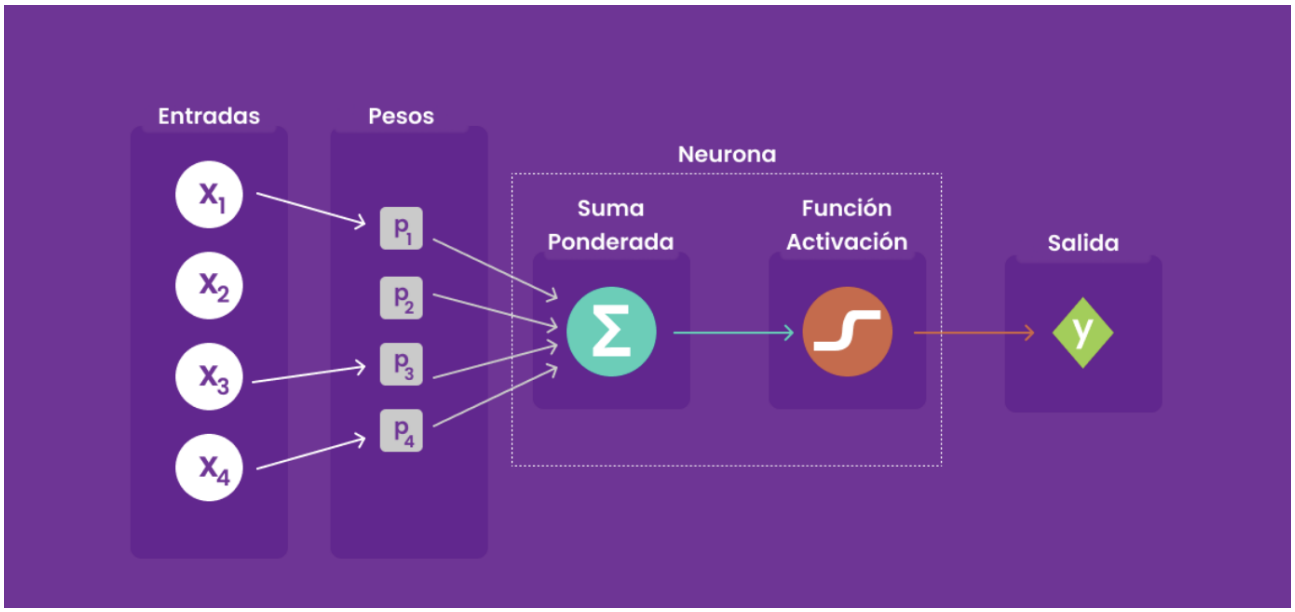

Figura 2.1: Diagrama Perceptrón

Los pesos que vemos en la figura anterior, se asignan inicialmente de manera aleatoria y se van reajustando con cada iteración de la red neuronal, asignando diferentes cantidades dependiendo de la importancia de la variable, hasta lograr una salida óptima, donde este reajuste, se realiza cuando no se supera el límite de decisión que activa la neurona.

La otra pieza clave dentro de las redes neuronales, es la función de activación, lo que activa la neurona, para dar una salida. Existen varias funciones de activación, en concreto la que utiliza el Perceptrón y la más básica es la función sigmoide, para problemas binarios, pero en rasgos generales, y en redes neuronales más avanzadas, se utiliza la función de activación ReLU [9], como referencia se toma la figura 2.2. Para poner más en contexto esta última función, tendremos que conocer las particularidades que la hacen la función de activación más utilizada, y esto es porque permite una mejor optimización mediante el descenso de gradiente estocástico [10] (*modelo de optimización para asignar mejores pesos a las variables*), un cálculo más eficiente e, independientemente del número de entradas, sus características no se ven afectadas.

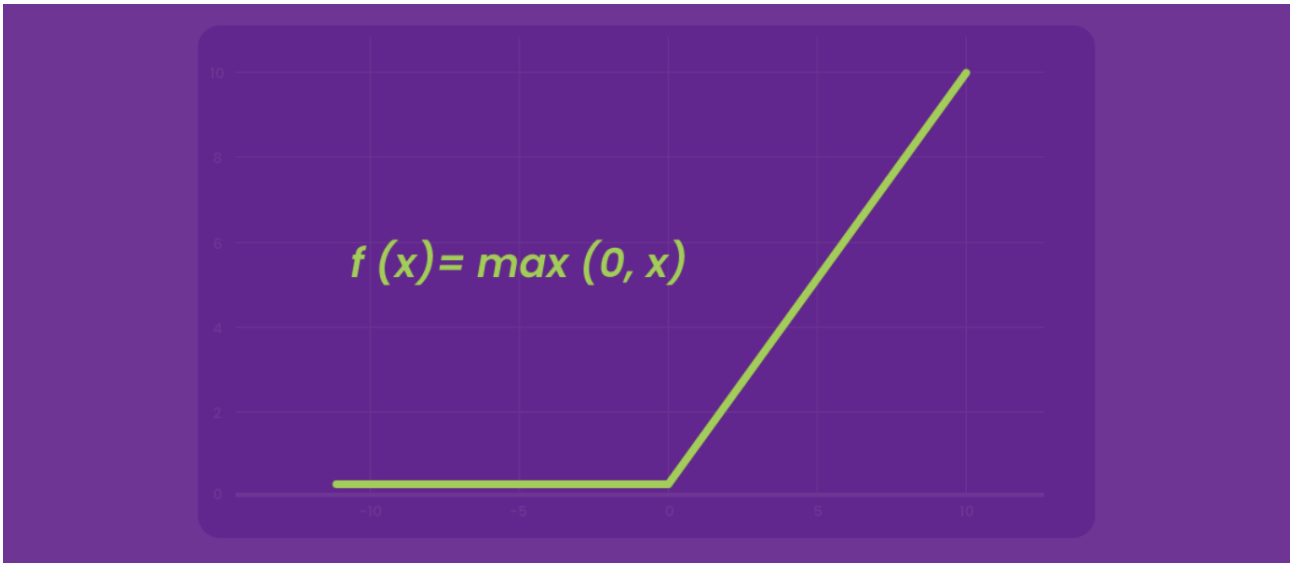

Figura 2.2: Función activación ReLU

La evolución de las redes neuronales a como las conocemos actualmente, se debe a la unión de varias capas de neuronas, donde cada capa tiene una cantidad de neuronas independientes de las otras capas, esto se conoce como red neuronal prealimentada, un ejemplo de esto es la figura 2.3. Cada capa alimenta a la siguiente con el resultado de su cálculo, su representación interna de los datos. Esto recorre todo el camino a través de las capas ocultas hasta la capa de salida, y es conocido como *retropropagación*.

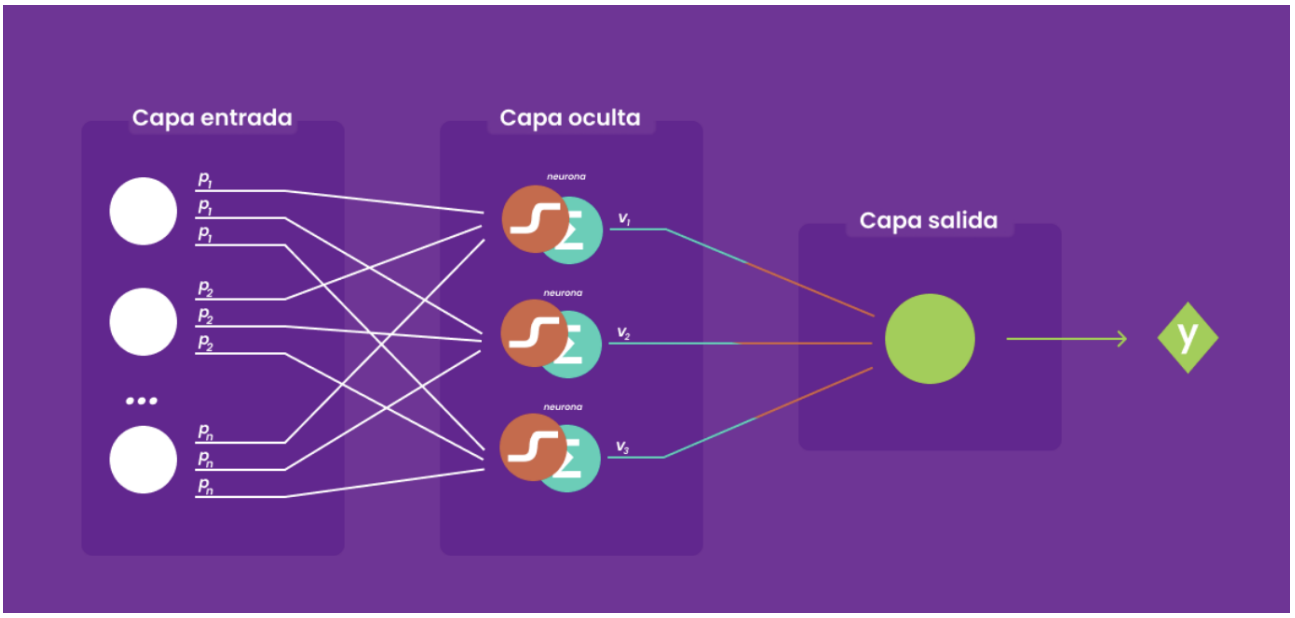

Figura 2.3: Diagrama Red Neuronal prealimentada

Redes Neuronales Recurrentes

Las redes neuronales recurrentes son un tipo de red pensada para tratar con datos secuenciales como series temporales, texto, secuencias de ADN, etc. Donde se procesan los datos de las secuencias elemento a elemento. Además, en cada momento también se tiene en cuenta información del momento anterior, es decir, almacenan un contexto (memoria).

Teniendo la figura 2.4 como referencia, podemos observar cómo se estructura la memoria en las redes neuronales recurrentes:

- Se estudian los estados ocultos h de la red neuronal recurrente, esto es equivalente a las capas ocultas es las redes neuronales pre-alimentadas, en un espacio de tiempo t, pero añadiendo una nueva variable que corresponderá al estado oculto del tiempo anterior, es decir, h_{t-1}.
- Posteriormente se realizan las mismas operaciones que son habituales en una neurona para producir una salida, utilizando la suma ponderada, la función de activación y el estado oculto anterior.

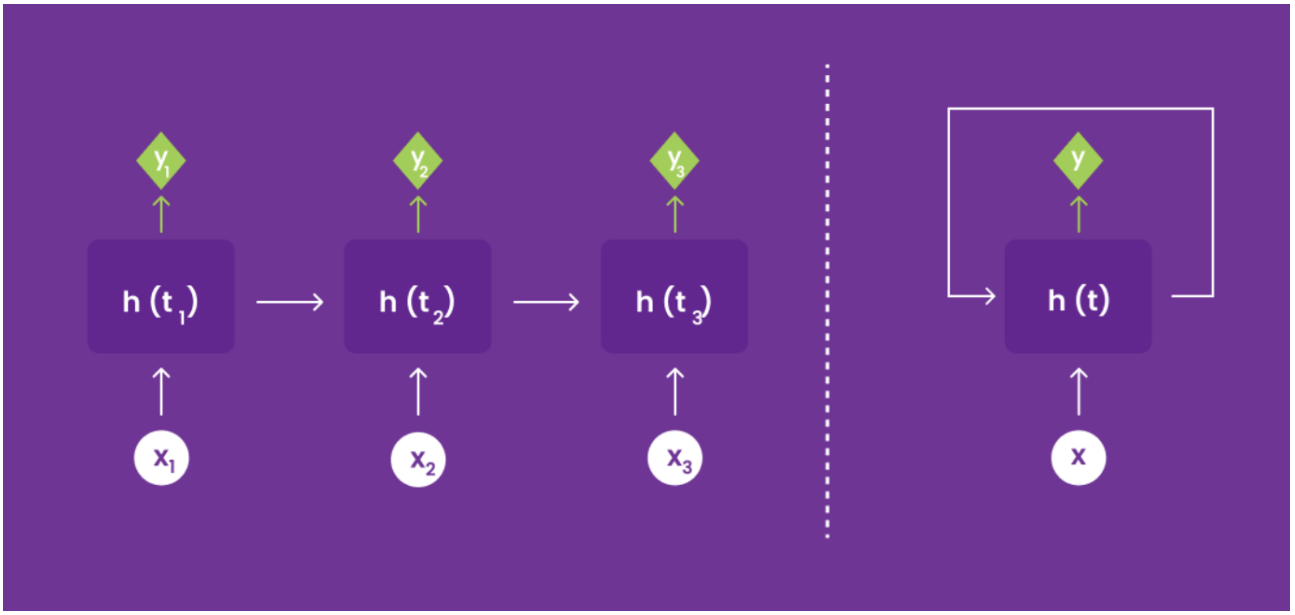

Figura 2.4: Diagrama Red Neuronal Recurrente

Long Short-Term Memory (LSTM)

Este tipo de red neuronal está basada en las redes neuronales recurrentes, pero, estas se diferencian en dos aspectos principales:

- Además del estado oculto, se encuentra un nuevo estado celda.

- Existe mayor cantidad de operaciones por elemento de la secuencia.

Las redes LSTM tienen unos objetivos algo diferentes a las RNN debido a los aspectos que comentamos anteriormente, estos objetivos son:

- **Olvidar el pasado irrelevante:** el vector f_t se obtiene al pasar un vector a través de una función sigmoide y multiplicarlo elemento a elemento con el estado de la celda anterior. Dado que la función sigmoide devuelve valores entre 0 y 1, se puede interpretar el vector f_t como un filtro que decide qué información se debe mantener o descartar en el estado anterior de la celda.
- **Guardar la información relevante del estado actual:** g_t calcula la información a añadir al estado de la celda. Existe un i_t igual que en el paso anterior para filtrar la información dejando la información más importante del estado actual.

Después de realizar estos dos objetivos nuevos, se vuelve a calcular el estado oculto el cual se genera como salida del estado actual y como entrada del estado posterior. En resumen, sabemos que las RNN sobrescriben su estado en cada iteración, mientras que con las redes LSTM su estado actual se calcula como una diferencia con el estado anterior.

2.2.3 Series Temporales

Las series temporales [13] son secuencias ordenadas cronológicamente de observaciones tomadas en intervalos regulares de tiempo. Se pueden identificar cuatro componentes diferentes en una serie temporal, como se muestra en la figura 2.5:

- La tendencia indica si la serie temporal tiene una tendencia ascendente o descendente a largo plazo.
- La variación estacional es un patrón recurrente que ocurre en momentos específicos, como ciertas épocas del año o días de la semana.
- La variación cíclica es una oscilación que no sigue un patrón fijo, y generalmente dura más de un año.
- El ruido hace referencia a las fluctuaciones aleatorias que no siguen ningún patrón o tendencia.

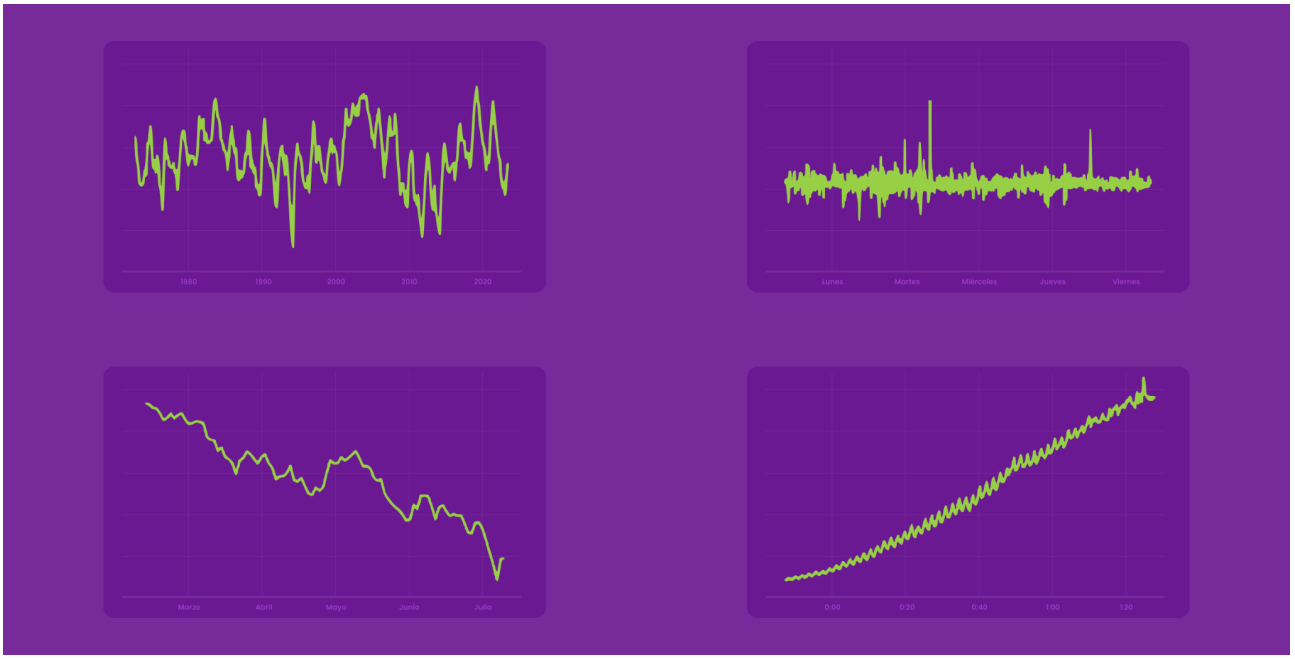

Figura 2.5: Ejemplos series temporales

Para poder entender con claridad los modelos de series temporales, tendremos que conocer el término *serie temporal estacionaria* [14]. Una serie temporal es estacionaria cuando las propiedades estadísticas como la media y la varianza no varían con el paso del tiempo, es decir, se mantienen constantes. La serie temporal no presenta cambios sistemáticos en el tiempo y es independiente del momento en que se observa.

Modelo serie temporal: ARIMA

El modelo ARIMA se utiliza para analizar y predecir valores futuros en una serie temporal estacionaria. En caso de que la serie no sea estacionaria, se puede aplicar una transformación para convertir dicha serie en estacionaria. La transformación más común es la diferenciación, la cual implica calcular la diferencia entre los valores consecutivos en la serie temporal. Dado que el tiempo se considera una variable discreta y que las observaciones se toman en intervalos regulares, la operación de diferenciación es equivalente a una derivada discreta.

El modelo ARIMA tiene tres componentes principales:

- **Autorregresivo (AR):** Es un modelo de regresión lineal que toma como variables de entrada los p valores anteriores de una serie temporal.
- **Integrada (I):** Representa la cantidad de veces que se ha aplicado la diferenciación a la serie temporal original para hacerla estacionaria, es decir, para eliminar la tendencia y estacionalidad de la serie.

- **Media móvil (MA):** Es un modelo de regresión que utiliza la media de la serie temporal como base para la predicción del siguiente punto. La media móvil se ajusta añadiendo una combinación lineal de los q errores anteriores de la serie temporal, donde q es el orden de la media móvil.

Partiendo del modelo ARIMA nos encontramos con dos extensiones que son un apartado clave dentro de las series temporales:

- **SARIMA*:*** Es una extensión del modelo ARIMA para manejar series temporales con variaciones estacionales. Esto se debe a que el modelo de diferenciación puede no estacionar completamente una serie temporal debido a sus variaciones.
- **SARIMAX*:*** Es una extensión del modelo SARIMA el cual está centrado en la utilización de nuevas variables a tener en cuenta para la predicción del modelo, incorporándose como otra autoregresión lineal.

Modelo serie temporal: Facebook Prophet

El modelo Facebook Prophet [15] está diseñado para proporcionar pronósticos precisos y escalables para series temporales de diferentes escalas. Este modelo se basa en un enfoque de descomposición aditiva que permite modelar la tendencia no lineal, la estacionalidad y los efectos de los días festivos de manera flexible. Este modelo también puede manejar valores faltantes y anomalías en los datos de la serie temporal, y ofrece la posibilidad de incluir regresores adicionales en el modelo.

Prophet modela la tendencia como una función no lineal que puede tener mayor o menor ruido, donde la estacionalidad se modela como una suma de funciones periódicas con diferentes periodos y amplitudes. Además, se pueden incluir regresores adicionales en el modelo para capturar otros factores que puedan influir en la serie temporal, como eventos importantes, características del mercado, etc. En general, el modelo Prophet ha demostrado ser eficaz para predecir series temporales en una amplia gama de aplicaciones, desde la previsión de ventas hasta la predicción de la calidad del aire.

El modelo Prophet, ha evolucionado hasta el punto de complementarse con modelos de Machine Learning para mejorar su precisión en las predicciones, de esto, nace Neural Prophet [16]. Este modelo utiliza una arquitectura de red neuronal recurrente para modelar la serie temporal y aprende automáticamente la mejor estructura de modelo para los datos. El modelo Neural Prophet también tiene en cuenta la incertidumbre en las predicciones mediante el uso de técnicas de inferencia Bayesiana, lo que permite proporcionar intervalos de confianza para los pronósticos. En general, el modelo Neural Prophet es un enfoque más avanzado y potente para la predicción de series temporales que puede manejar mejor la complejidad y la incertidumbre en los datos de la serie temporal.

2.3 Modelo de lenguaje para la generación de texto

Los modelos de generación de texto, son modelos de lenguaje basados en redes neuronales, los cuales se caracterizan por la generación de texto coherente y semánticamente relevante a partir de una entrada específica, donde a esta entrada se le denomina *prompt* [18].

Estos modelos, utilizan técnicas de aprendizaje profundo o Deep *Learning*, y una gran cantidad de datos de calidad para poder realizar un entrenamiento lo suficientemente robusto como para aprender las relaciones entre las palabras y la estructura gramatical de nuestro lenguaje. Entre los modelos de generación de texto más avanzados encontramos, text-davinci-003, text-curie-002, GPT-3, GPT-3.5-turbo y el más reciente y potente de todos, GPT-4 [19]. Los tres últimos modelos comentados no solo tienen la función de generar texto nuevo, sino de tener la funcionalidad de chat, es decir, de poder ir almacenando un contexto de la conversación que se está teniendo con el modelo. Esto hace que puedas hacer referencia a cosas anteriormente comentadas o pedir alguna alternativa a los textos que te está generando.

Estos modelos tienen millones de parámetros y son capaces de generar texto que es difícil de distinguir del texto escrito por un humano. Además, estos modelos pueden realizar tareas específicas, como la traducción de idiomas, la generación de resúmenes y la generación de respuestas a preguntas.

2.4 Chatbots

Los chatbots están diseñados para interactuar con los usuarios a través de conversaciones en lenguaje natural. Estos sistemas utilizan técnicas de machine learning como el Procesamiento del Lenguaje Natural [20] (Natural Language Processing en inglés) para entender la intención del usuario y proporcionar una respuesta adecuada a la solicitud. En particular, los chatbots de Telegram son una forma popular y open source de chatbot que se utilizan para interactuar con los usuarios a través de un chat de la propia plataforma de mensajería instantánea.

Estos chatbots se basan en el uso de la API de Telegram [21] para recibir y enviar mensajes. Como normal general estos bots se programan utilizando lenguajes de programación como Python y se alojan en servidores que están en línea y accesibles para los usuarios. Los chatbots no solo se utilizan como una herramienta para el uso de técnicas como la clasificación de intenciones, la identificación de entidades, etc, sino también para generar una interfaz amigable, rápida y fácil de usar para los usuarios.

En general, los chatbots de Telegram son una herramienta útil y efectiva para interactuar con los usuarios y proporcionar una experiencia de usuario de alta calidad.

Capítulo 3 Tecnologías y arquitectura de la herramienta

3.1 Tecnologías

3.1.1 Base de datos

El software con el que se trata la información de TITSA y dónde se almacenan los datos es un SQL Server de Microsoft [22]. Para poder simular este acceso y dejar la herramienta lo más adaptada posible al entorno TITSA se ha decidido recrear una base de datos SQL Server de manera local, además gracias a la verificación utilizando las propias credenciales de windows, el acceso a la base de datos es mucho más rápida y eficiente.

3.1.2 Tratamiento de datos

Para la realización del proyecto se ha decidido optar por el lenguaje de programación Python. Esto se debe a que a día de hoy es el lenguaje de programación por excelencia para la resolución de problemas como el que estamos tratando. Otro punto a destacar, es la familiaridad con el lenguaje de programación, su comunidad y la gran versatilidad que tiene.

En concreto, el tratamiento de datos, su entrenamiento y posterior evaluación se ha realizado con las siguientes librerías:

- **Pandas**, **Numpy** y **Plotly**, para la carga, manipulación y visualización de los datos.
- **Preln** para el preprocesamiento de los datos en lenguaje natural.
- **Statsmodels** para los modelos ARIMA, SARIMA y SARIMAX.
- **neuralprophet** para el modelo Neural Prophet.
- **tensorflow / keras** para los modelos de redes neuronales como LSTM.
- **sklearn** para las métricas de evaluación de los modelos.
- **telegram-python-bot** para la creación del bot de Telegram que se usará de interfaz.
- **sqlalchemy** para la conexión con la base de datos y realización de las consultas.
- **openai** para la utilización de los modelos de generación de texto.

Entre otras librerías más secundarias que se utilizarán para poder realizar el proyecto de la forma más profesional, escalable y limpia posible. El código del proyecto tiene una estructura robusta y completa, la cual engloba las diferentes fases del proyecto y las integra en un único directorio almacenado en un controlador de versiones como GitHub [23].

3.2 Estructura de la herramienta

Para poder analizar en detalle la estructura de la herramienta que se ha desarrollado, se ha decidido realizar diversos diagramas para tener una visión global de cada fase importante del proyecto. En la figura 3.1 se puede observar la estructura global del proyecto.

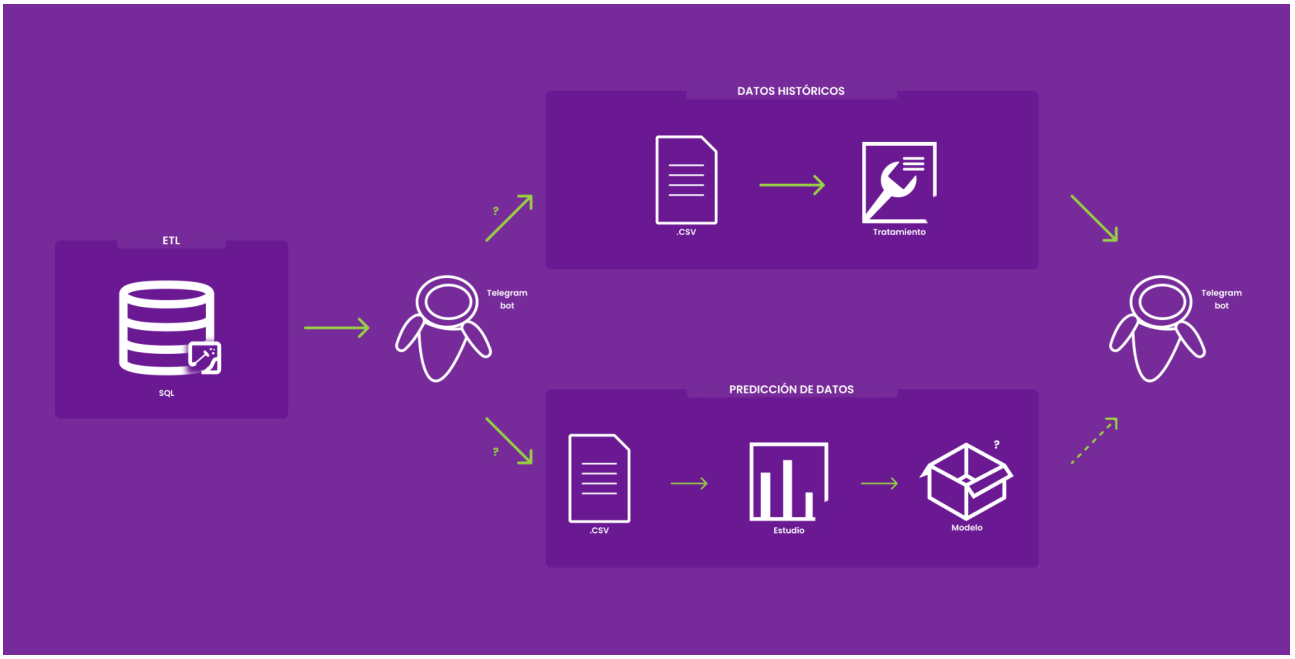

Figura 3.1: Estructura global del proyecto

A continuación revisaremos el diagrama general del modelo ETL de la herramienta. A lo largo del capítulo 4, se explicarán los diagramas para cada una de las fases del proyecto.

3.2.1 Diagrama ETL

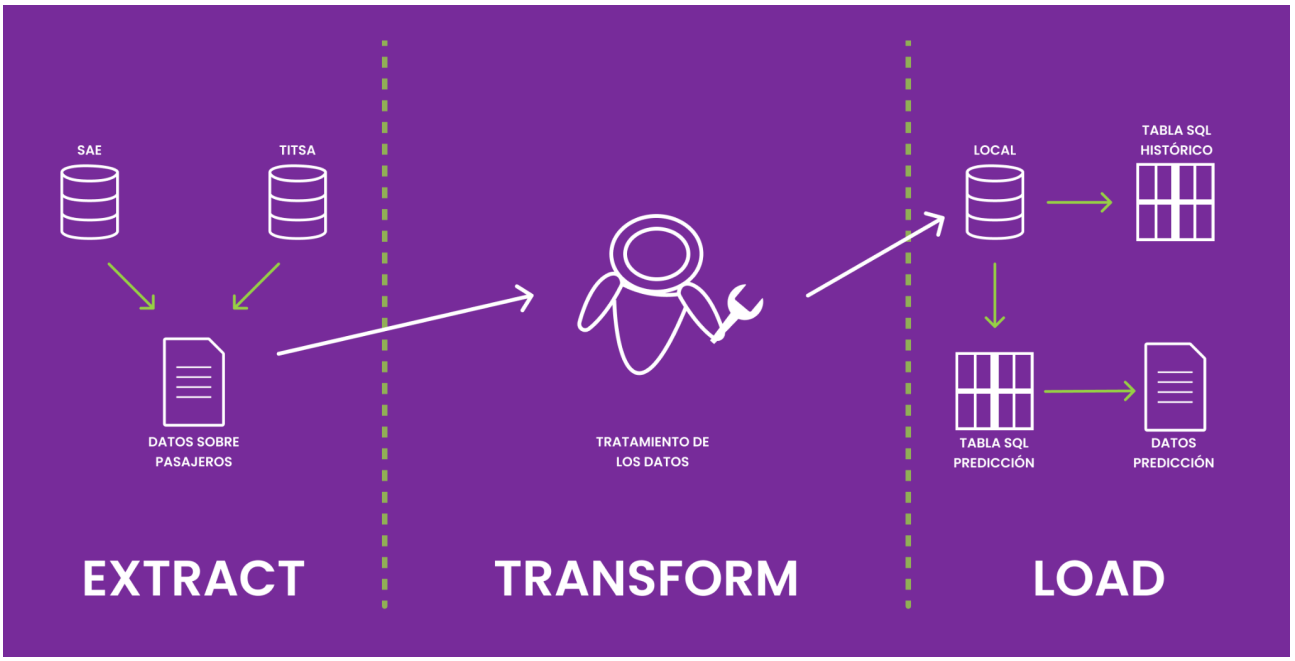

Figura 3.2: Estructura ETL

Un proceso ETL como el que podemos visualizar en la figura anterior, consiste en:

- **Extract:** Extraer datos de uno o varios orígenes.
- **Transform:** Transformar los datos según sea conveniente, aplicando técnicas como la limpieza, combinación o eliminación de datos.
- **Load:** Cargar los datos transformados en una nueva fuente de datos.

Para el presente proyecto, se tuvo que realizar un ETL y así contar con unos datos de calidad tanto para la herramienta de consulta a datos históricos, como para realizar el modelo de predicción de datos. En rasgos generales, para la herramienta de consulta a datos históricos se trabajará directamente desde la tabla sql realizando consultas a través de scripts. En cambio, para poder realizar un correcto análisis y entrenamiento de los datos necesarios para la predicción, se ha decidido extraer los datos en formato csv desde la base de datos local.

Entre las técnicas de transformación de los datos, se encuentran:

- Eliminación de las columnas innecesarias y las filas con valores erróneos.
- Formateo de los nombres de las columnas o de los valores internos para una correcta interpretación.
- Combinación de columnas como la fecha y la hora que serán de utilidad para el modelo de predicción.
- Conversión de los tipos de datos para un mejor manejo de los valores.

Capítulo 4 Desarrollo

4.1 Descripción de los datos

Para este proyecto se ha trabajado con dos estructuras diferentes de datos. La correspondiente a la primera fase (acceso a datos históricos) maneja los datos históricos sobre la cantidad de pasajeros diarios, divididos por línea, y con un intervalo de tiempo que agrupa los dos últimos años. En la figura 4.1 se puede apreciar un extracto de este conjunto de datos para ver su composición.

	FECHA	TIPO_DIA	LINEA	ZONA	TIPO_CONVENIO	PRODUCT_CODE	PASAJEROS
1	2022-11-17	Jueves	901	URBANO SC	URBANO SC	88	6
2	2022-11-17	Jueves	901	URBANO SC	URBANO SC	150	3
3	2022-11-17	Jueves	901	URBANO SC	URBANO SC	170	9
4	2022-11-17	Jueves	901	URBANO SC	URBANO SC	171	3
5	2022-11-17	Jueves	901	URBANO SC	URBANO SC	172	1
6	2022-11-17	Jueves	901	URBANO SC	URBANO SC	200	238

Figura 4.1: Base de datos, tabla datos históricos

Estos datos son extraídos por parte del departamento de Big Data & Data Science de TITSA en formato csv. Con el fin de replicar el ecosistema de la empresa y dejar la herramienta lo más adaptada posible a ese ecosistema, se ha decidido replicar la base de datos de manera local y almacenar en ella los datos correspondientes a la cantidad de pasajeros de los últimos dos años. De esta forma se podrán realizar consultas directamente sin tener que depender de la extracción de un fichero.

Con respecto a la segunda fase (predicción de pasajeros), los datos también se han obtenido en formato csv y se han replicado en una base de datos local para su extracción desde un script de Python. Como se puede observar en la figura 4.2, la cantidad de columnas y filas redundantes e innecesarias para realizar un modelo de predicción a nivel teórico es elevado, por lo que se ha decidido realizar un análisis y se ha construído un nuevo conjunto más refinado, con las columnas necesarias y la cantidad de datos correspondientes al primer mes de 2023. Esto es debido a que enero es el primer mes tras la aplicación de las nuevas tarifas de transporte y, si se realiza un modelo con datos referentes a una tarifa anterior, podría dar pie a valores irreales.

Results Messages

	PLN_DIAPLANIFICADO	PLN_IDEXPEDICION	PLN_EXP_HHSALIDA	PLN_EXP_IDITINERARIO	PLN_ITI_CODIGO	PLN_ITI_IDSENTIDO	PLN_EXP_IDPARADA_ORIGEN	PLN_EXP_IDPARADA_DESTINO	PLN_EXP_SALIDA_OPERATIVA	PLN_EXP_LLEGADA_OPERATIVA	PLI
1	2022-12-12 00:00:00.000	5415667	2325	2601	131	1	131	3885	2022-12-12 23:25:00.000	2022-12-13 00:10:00.000	33
2	2022-09-27 00:00:00.000	5168206	1420	1495	11	1	131	2176	2022-09-27 14:20:00.000	2022-09-27 15:30:00.000	79
3	2022-11-25 00:00:00.000	5367317	1545	2644	31	1	131	2176	2022-11-25 15:45:00.000	2022-11-25 16:50:00.000	75
4	2022-12-18 00:00:00.000	5449099	1110	2532	112	2	3885	188	2022-12-18 11:10:00.000	2022-12-18 12:05:00.000	32
5	2022-11-30 00:00:00.000	5391032	1235	2532	112	2	3885	188	2022-11-30 12:35:00.000	2022-11-30 13:25:00.000	32
6	2022-10-02 00:00:00.000	5204385	1945	1495	11	1	131	2176	2022-10-02 19:45:00.000	2022-10-02 20:50:00.000	79
7	2022-09-22 00:00:00.000	5141729	1600	2531	111	1	131	3885	2022-09-22 16:00:00.000	2022-09-22 16:55:00.000	33

Figura 4.2: Base de datos, tabla datos predicción

4.2 Herramienta para el acceso a datos históricos

Apoyándonos en la figura 4.3 la primera fase tiene un alcance que comprende el desarrollo de:

- Sistema de filtrado para la generación de consultas.
- Sistema para la generación de consultas a través de lenguaje natural.
- Generación de una respuesta elaborada haciendo uso del modelo de generación de texto.
- Control de usuarios, para que varios usuarios puedan utilizar el bot simultáneamente.
- Control de errores, devolviendo un feedback constante al usuario en caso de que se produzca algún error interno de la herramienta o referente a los datos.

Los actores implicados en la interacción son el usuario, el bot de Telegram y el modelo de generación de texto (GPT). El usuario y el bot de Telegram son actores principales y mantienen una conversación, mientras que el modelo de generación de texto funciona como actor secundario y se utiliza en momentos específicos de la conversación.

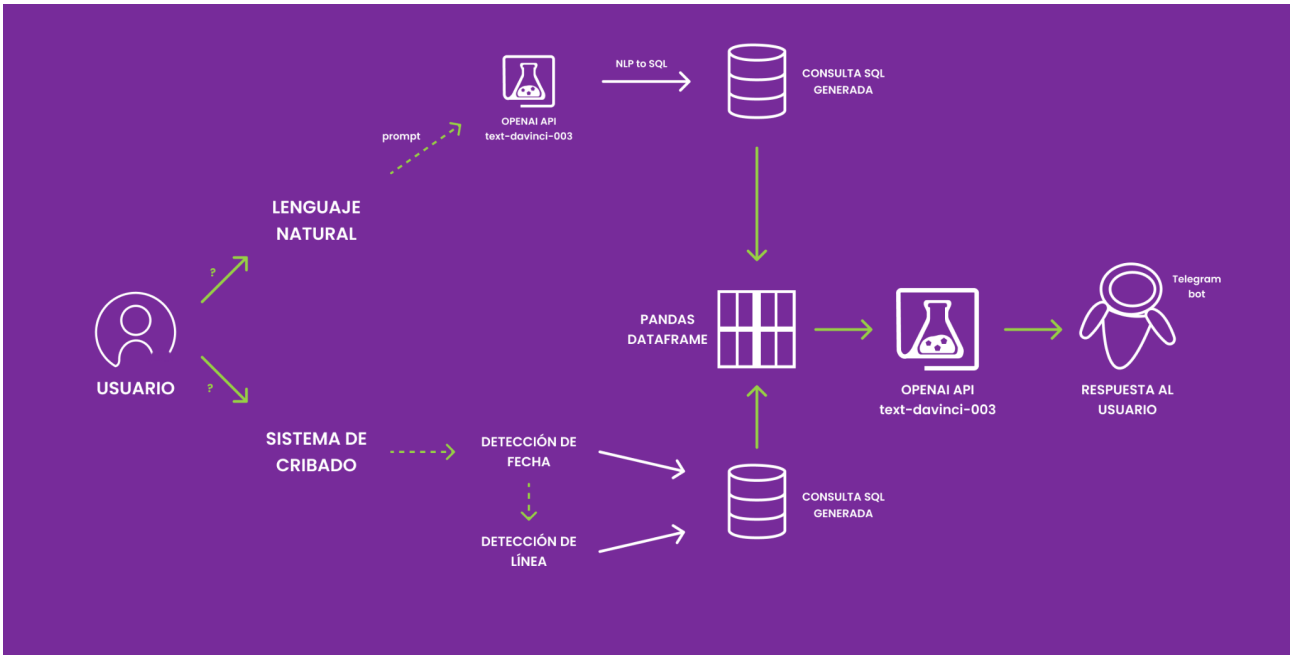

Figura 4.3: Estructura consulta de datos históricos

4.2.1 Creación del bot de Telegram

Para poder construir un bot de Telegram, es necesario tener un token de autenticación. Este token se consigue a través de un bot oficial de la plataforma, llamado Bot Father [24], que se encarga de la generación y gestión de un nuevo bot. Una vez se ha creado el bot, asignado un nombre y un usuario, se genera el token de autenticación, el cual se utilizará para poder acceder a la API de Telegram Bot.

Existen diversas formas de acceder a los métodos de la API de Telegram y construir un bot en la plataforma. No obstante, en el caso de este proyecto, se optó por la utilización de una librería específica desarrollada en el lenguaje de programación Python, denominada *python-telegram-bot*. Dicha librería es una de las más utilizadas en la creación de bots de Telegram y se caracteriza por proporcionar una interfaz puramente asíncrona de Python que otorga acceso a la API de Telegram Bot.

La librería *python-telegram-bot* presenta una amplia variedad de métodos que permiten llevar a cabo diversas acciones en el bot de manera eficiente y simplificada. Con ello, se facilita la creación y el manejo de bots en la plataforma de Telegram de manera ágil y efectiva.

4.2.2 Estructura y funcionamiento de la herramienta

Fijándonos en la figura 4.3, observamos que al usuario cuando inicia una conversación con el bot se le presentan dos opciones, la primera hace referencia a consultar los datos históricos a través de un filtrado y la segunda es realizar la misma consulta, pero directamente con lenguaje natural.

Teniendo en cuenta el orden cronológico de la creación de ambos métodos, pasaremos a ver primero la consulta de datos a través de un filtrado.

Consulta de datos históricos a través del sistema de filtrado

En un primer momento, se nos presentó el problema de recoger a través de lenguaje natural la fecha o el rango de fechas que el usuario desea comprobar, como se puede apreciar en la figura 4.4.

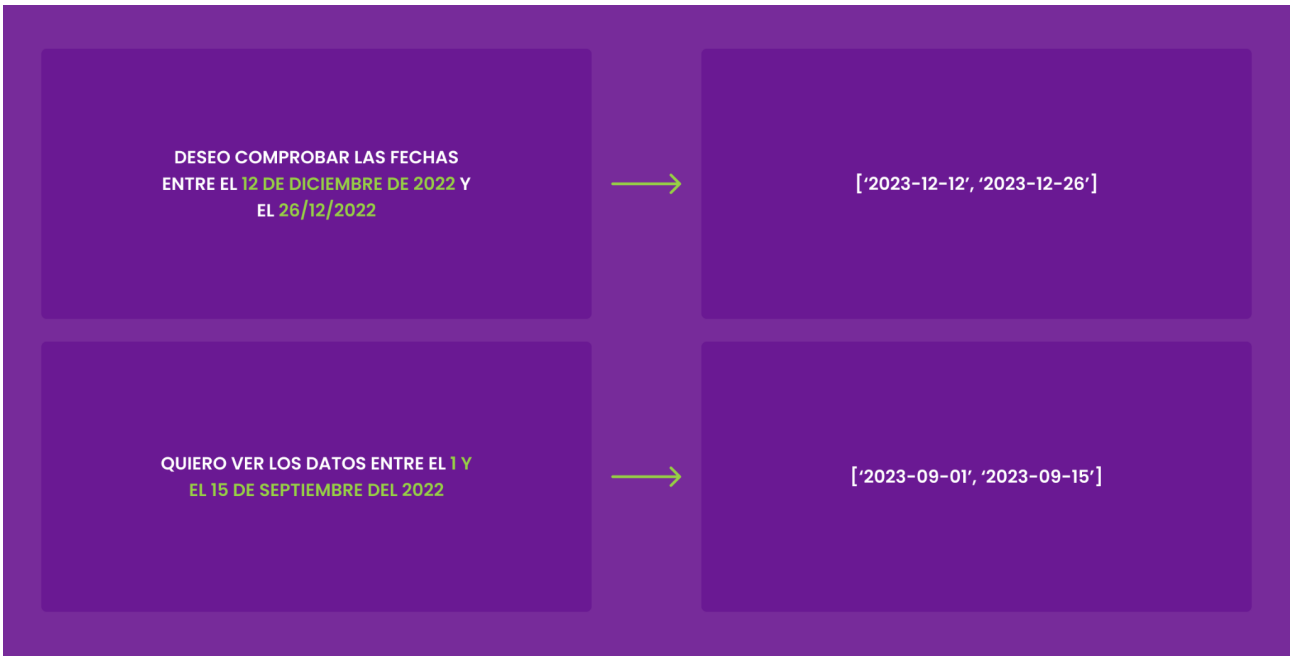

Figura 4.4: Ejemplo detección de fechas

Inicialmente se consideró el uso de modelos de Machine Learning enfocados en el procesamiento del lenguaje natural para abordar el problema de la detección de fechas en textos. Sin embargo, tras un análisis más detallado, se evidenció que los modelos pre-entrenados disponibles no brindaban resultados óptimos para solucionar el problema. Esto se debía a que la detección de fechas dependía en gran medida del formato en el que se ingresaba la información, lo que dificultaba el procesamiento preciso y eficiente de los datos.

Además, se evaluó la posibilidad de entrenar un modelo propio a partir de un modelo pre-entrenado, sin embargo, la cantidad y calidad de datos disponibles no eran suficientes para llevar a cabo un entrenamiento efectivo. Por lo tanto, se decidió buscar alternativas para solucionar el problema de detección de fechas en textos en el contexto del desarrollo del bot de Telegram.

Finalmente, se optó por hacer uso de expresiones regulares, las cuales dieron unos resultados excelentes. A la expresión regular que se utilizó se puede acceder a través del *repositorio GitHub* y engloba todos los casos que se muestran en la figura 4.4.

Continuando con el sistema de filtrado, para la detección de las líneas de transporte se optó por las expresiones regulares, teniendo en cuenta los buenos resultados que se han obtenido con la detección de las fechas. A parte de las expresiones regulares, se añadieron algunas reglas como:

- **Todas:** Referencia a todas las líneas de transporte.
- **Largo recorrido:** Referencia a las líneas de largo recorrido como la 110 o la 111.
- **Urbano:** Referencia a las líneas urbanas.

Esto ayuda al usuario a no tener que escribir todas las líneas manualmente, y agilizar el proceso de búsqueda.

Una vez se han obtenido el rango de fecha y las líneas de las que se quieren obtener los datos, se autorellena una plantilla de una consulta SQL previamente diseñada para obtener los datos correspondientes. Una vez se rellena esta plantilla, y haciendo uso de la librería sqlalchemy se realiza la consulta, se obtienen los datos y se transforman en un dataframe de pandas para poder manipular los datos de una manera sencilla y eficiente.

La herramienta cuenta con diversas plantillas, abarcando todas las posibles opciones que tiene el usuario, fecha única, rango de fechas, una o varias líneas de transporte, etc.

Consulta de datos históricos a través del lenguaje natural

Con el objetivo de conseguir una herramienta lo más accesible y completa posible, se volvió a estudiar la posibilidad de usar modelos de lenguaje natural para la obtención del rango de fechas y las líneas de transporte, pero, en lugar de utilizar un sistema de filtrado, se presentó la posibilidad de tratar el texto de entrada del usuario como un medio para transformar el lenguaje natural a una consulta SQL.

Tras investigar esta técnica, nos encontramos con que la mayoría de los modelos centrados en NLP to SQL están en fase de investigación, por lo que, solo se cuenta con artículos (papers) que tratan los avances teóricos de dichos modelos. No obstante, se cuenta con el modelo de generación de texto más revolucionario hasta la fecha, GPT, creado por OpenAI. Viendo sus prestaciones se probó la transformación de un prompt o texto de entrada que haría un usuario a una consulta SQL la cual sirva para la obtención de los datos. Tras varias pruebas, conocimiento de ingeniería del prompt y algunas reglas, se obtuvo un buen resultado, pudiendo realizar consultas SQL de bajo nivel directamente con lenguaje natural.

Poniendo en contexto al modelo de generación de texto, en nuestro caso, text-davinci-003, pasándole como parámetro el prompt del usuario, y ajustando los hiper parámetros, conseguimos resultados como el que se visualiza en la figura 4.5.

Figura 4.5: Ejemplo consulta NLP to SQL

Al igual que en el caso anterior, los datos obtenidos se transforman en un dataframe de pandas con el objetivo de manipular los datos de manera correcta. Una vez se obtienen los datos que desea consultar el usuario, se buscó una forma de dar una respuesta elaborada y no ceñirnos a escribir los datos directamente.

Viendo los buenos resultados que ha dado el modelo de generación de texto, se propuso emplear el mismo modelo, pero con un objetivo diferente para elaborar una respuesta al usuario lo más "humanamente posible" sin hacer uso de una estructura predeterminada. Tras varias pruebas buscando el mejor prompt posible que generase una respuesta con sentido y precisa, se consiguieron resultados como el que se aprecian en la figura 4.6. Se puede observar como para exactamente el mismo mensaje de entrada por el usuario, ya sea por lenguaje natural o por sistema de filtrado, se generan respuestas totalmente diferentes, y alguna de ellas puede sorprender.

Figura 4.6: Ejemplo respuesta al usuario

Dificultades encontradas

Durante el desarrollo del bot de Telegram, se encontraron varias dificultades que entorpecieron el proceso de programación y afectaron al rendimiento del mismo. La primera dificultad encontrada fue la falta de documentación adecuada en la librería python-telegram-bot, lo que obligó a basar gran parte del trabajo en pruebas y observaciones del comportamiento de la herramienta. Esto hizo que el proceso de desarrollo fuera más lento y complicado de lo esperado.

Además, se encontró que el bot se colgaba automáticamente en caso de fallos internos, lo que generó un impacto negativo en la experiencia del usuario. Para resolver este problema, se programó un control de errores genérico y específico que proporciona un feedback constante al usuario y evita que el bot se cuelgue. Otra dificultad encontrada fue la falta de soporte por defecto para varios usuarios en la librería. Para superar esta limitación, se creó una tabla hash que almacenaba el ID de cada conversación y de cada mensaje, lo que permitió una gestión más efectiva de las respuestas y evitó la superposición de respuestas de usuarios diferentes.

Finalmente, se encontraron limitaciones en los modelos de OpenAI, debido a la disponibilidad limitada de tokens en la API. Por esta razón, se tuvo que ser muy preciso en la generación de las respuestas y en la formulación del prompt. Esto significó que se tuvo que invertir mucho tiempo en el diseño y la ingeniería del prompt para asegurar la calidad de las respuestas generadas. En resumen, estas dificultades encontradas en el desarrollo del bot de Telegram obligaron a los desarrolladores a adoptar un enfoque más cuidadoso y riguroso para garantizar la calidad del producto final.

4.3 Modelo de predicción de pasajeros a futuro

4.3.1 Estudio y tratamiento de los datos

Para llevar a cabo la implementación de este modelo predictivo, se ha optado por utilizar como muestra los datos correspondientes al mes de enero de 2023. Esta elección se fundamenta en que dicho mes marca el inicio de la aplicación de las nuevas tarifas para los abonos de viaje de la empresa. Asimismo, con el propósito de realizar una estimación inicial, se ha seleccionado la línea de transporte 110 en dirección de Santa Cruz a Costa Adeje. En la figura 4.7 se puede observar el resultado de la extracción de este conjunto de datos.

	FECHA	LINEA	SENTIDO	TRAYECTO	PARADA	DIA	IS_WEEKEND	PASAJEROS
0	2023-01-01 08:45:00	110	1	31	9181	Domingo	1	11
1	2023-01-01 09:15:00	110	1	31	9181	Domingo	1	22
2	2023-01-01 09:45:00	110	1	31	9181	Domingo	1	9
3	2023-01-01 10:15:00	110	1	31	9181	Domingo	1	16
4	2023-01-01 10:45:00	110	1	31	9181	Domingo	1	28
...	...	...	...	...	...	...	...	...
747	2023-01-31 19:15:00	110	1	31	9181	Martes	0	21
748	2023-01-31 19:45:00	110	1	31	9181	Martes	0	15
749	2023-01-31 20:15:00	110	1	31	9181	Martes	0	28
750	2023-01-31 20:45:00	110	1	31	9181	Martes	0	26
751	2023-01-31 21:15:00	110	1	31	9181	Martes	0	20

752 rows × 8 columns

Figura 4.7: Dataframe línea de transporte 110

El horario de operación de los servicios de la línea de transporte 110 abarca desde las 5:45 hasta las 21:15 todos los días. Esto implica que, para el mes de enero, se dispone de un total de 992 registros de datos distintos. Sin embargo, al examinar la figura anterior y realizar un análisis detallado, se observa la presencia de valores faltantes, con un total de 240 casos. Estos valores ausentes representan aproximadamente un tercio del conjunto de datos completo necesario para llevar a cabo el experimento.

Debido a que aplicamos modelos de series temporales, es crucial contar con una línea de tiempo lo más completa posible. Por lo tanto, tenemos que rellenar los valores faltantes para poder obtener unos datos lo más afinados posibles. Para rellenar dichos valores, se ha decidido realizar una media con la cantidad de pasajeros del resto de días para la misma hora.

Como resultado de este procedimiento, y la inclusión de una nueva columna que verifica si en algún día de enero ha habido alguna festividad, obtenemos un conjunto de datos como el que se muestra en la figura 4.8.

	FECHA	LINEA	SENTIDO	TRAYECTO	PARADA	DIA	IS_WEEKEND	PASAJEROS	IS_HOLIDAY
0	2023-01-01 05:45:00	NaN	NaN	NaN	NaN	Sunday	1	37	0
1	2023-01-01 06:15:00	NaN	NaN	NaN	NaN	Sunday	1	27	0
2	2023-01-01 06:45:00	NaN	NaN	NaN	NaN	Sunday	1	20	0
3	2023-01-01 07:15:00	NaN	NaN	NaN	NaN	Sunday	1	23	0
4	2023-01-01 07:45:00	NaN	NaN	NaN	NaN	Sunday	1	19	0
...	...	...	...	...	...	...	...	...	...
991	2023-01-31 19:15:00	110.0	1.0	31.0	9181.0	Martes	0	21	0
992	2023-01-31 19:45:00	110.0	1.0	31.0	9181.0	Martes	0	15	0
993	2023-01-31 20:15:00	110.0	1.0	31.0	9181.0	Martes	0	28	0
994	2023-01-31 20:45:00	110.0	1.0	31.0	9181.0	Martes	0	26	0
995	2023-01-31 21:15:00	110.0	1.0	31.0	9181.0	Martes	0	20	0

992 rows × 9 columns

Figura 4.8: Dataframe completo línea de transporte 110

Dado que los valores correspondientes a las columnas línea, sentido, trayecto y parada son estáticos, no es necesario rellenarlos ya que no se tendrán en cuenta para la predicción del modelo. Hay que destacar que la parada seleccionada es la primera del sentido Santa Cruz a Costa Adeje, lo que implica que nuestra serie temporal sólo tendrá como dependencia los valores de los días anteriores en esa misma hora. En el caso de seleccionar una parada intermedia, sería necesario retroalimentar dicha serie temporal con el valor de la serie temporal en el instante $t - 1$, es decir, la parada anterior.

Con el objetivo de visualizar de manera más efectiva estos datos, se ha generado un histograma de puntos, como se muestra en la figura 4.9. Este histograma nos permite observar las variaciones en la cantidad de pasajeros para cada una de las horas establecidas, teniendo en cuenta los valores faltantes que han sido completados. Sin embargo, para analizar la concentración de valores, identificar valores fuera de rango, conocer la mediana, y examinar la tendencia, es más apropiado realizar un diagrama de cajas y bigotes como el que se muestra en la figura 4.10.

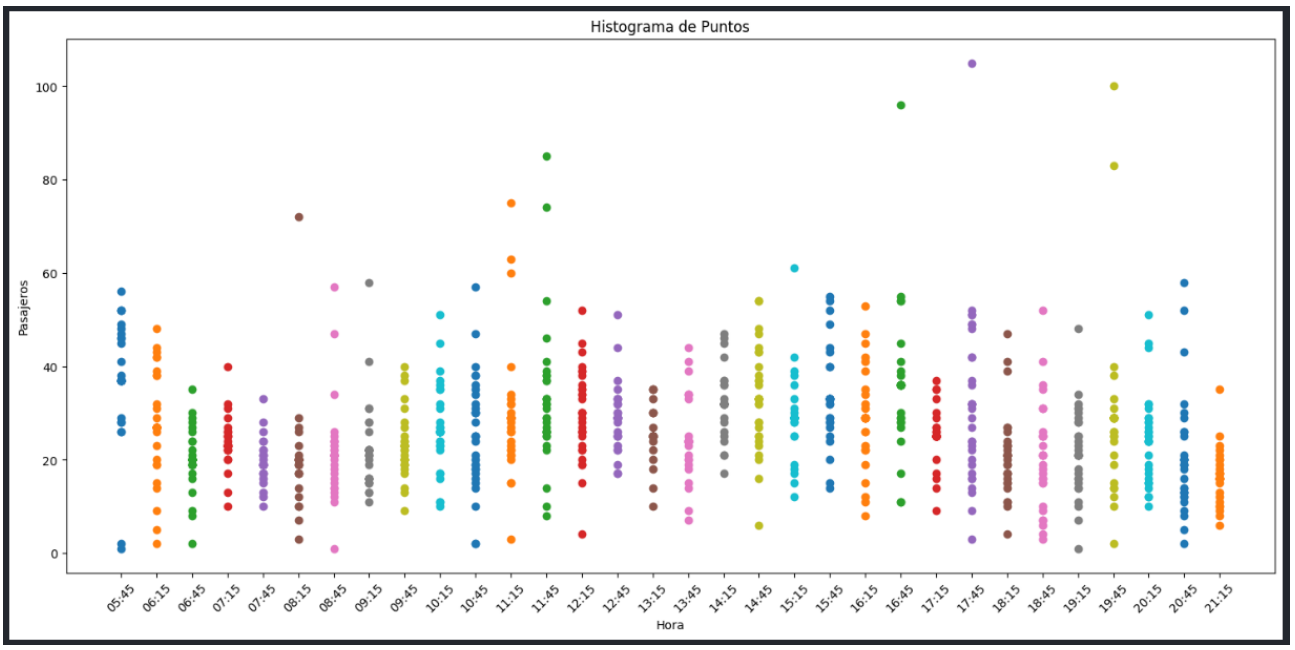

Figura 4.9: Histograma

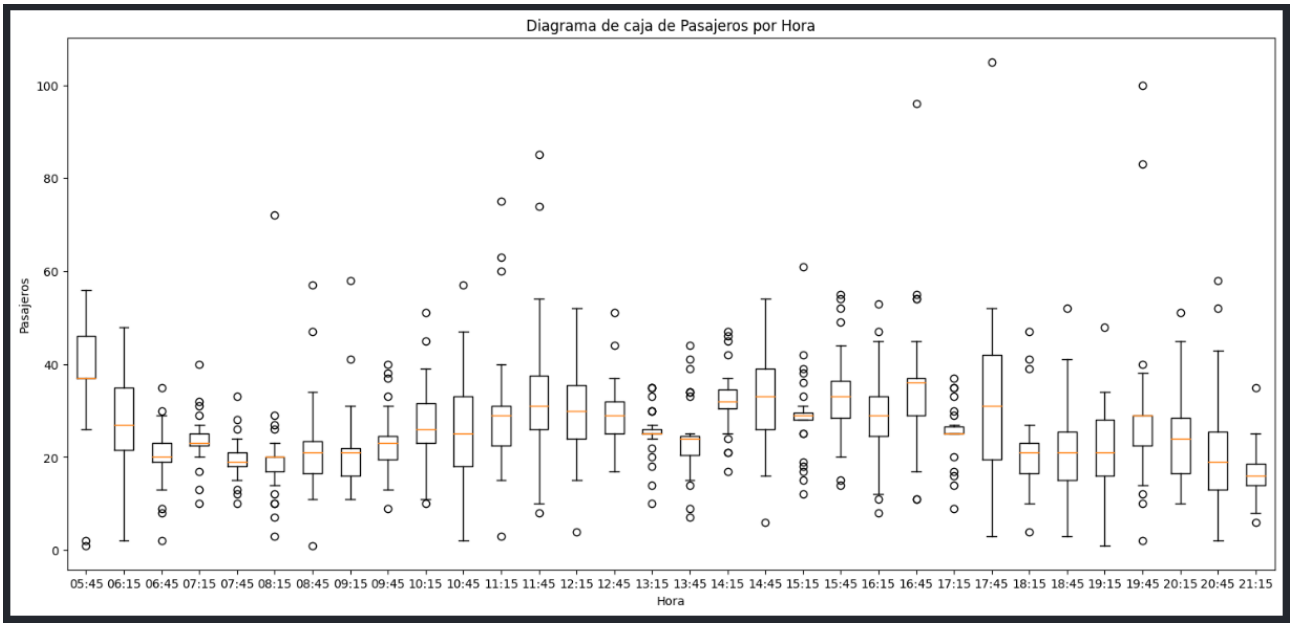

Figura 4.10: Diagrama de cajas y bigotes

En el diagrama de la figura 4.10, se puede apreciar que en ciertas horas pico, como las previas al inicio de la jornada laboral y las posteriores a su finalización, la caja se vuelve significativamente más estrecha. Este estrechamiento indica que en un gran número de días diferentes se registra una alta concentración de pasajeros durante esas horas específicas.

4.3.2 Métricas de evaluación de los modelos

Con el fin de evaluar el desempeño de los modelos, se han empleado las siguientes métricas en relación al conjunto de validación y las predicciones correspondientes a la última semana de enero:

- **Error absoluto medio:** es una métrica sencilla utilizada para evaluar la precisión de un modelo. Se calcula tomando la diferencia absoluta entre los valores predichos y los valores reales, promediando dichos valores para obtener una media global de error. Cuanto menor sea el valor de MAE, mayor será la precisión de nuestro modelo.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - f(x_i)\right|$$

- **Error cuadrático medio:** es una métrica utilizada para evaluar el rendimiento de un modelo. Se calcula tomando la diferencia al cuadrado entre los valores predichos y los valores reales, promediando dichos errores cuadráticos. El MSE proporciona una medida de la dispersión o variabilidad de las predicciones del modelo con respecto a los valores reales. Sin embargo, la desventaja del MSE es que las unidades son difíciles de interpretar.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - f(x_i)\right)^2$$

- **Raíz cuadrada del error cuadrático medio:** es una métrica utilizada para mejorar la interpretación de los valores del error cuadrático medio, esto se consigue realizando la raíz cuadrada de dichos valores, de esta forma se obtienen las mismas unidades que los valores de entrada.

$$RMSE = \sqrt{MSE}$$

- **Coeficiente de determinación:** es una métrica que relaciona la variabilidad de los datos alrededor del modelo con respecto a la variabilidad alrededor de la media. En caso de que el modelo prediga de manera precisa todos los valores, el resultado del coeficiente de determinación sería 1. Por otro lado, si el modelo es tan inexacto que la media de los datos proporciona una mejor predicción, el resultado sería 0 o incluso negativo.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left(y_i - f(x_i)\right)^2}{\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2}$$

Estas métricas son convencionales en la evaluación de modelos de regresión. No obstante, dada la escasez de datos disponibles en esta fase, existe la posibilidad de que estas métricas carezcan de fiabilidad. Por consiguiente, además de considerar estas métricas para determinar la calidad de un modelo, se otorga una atención particular a las visualizaciones que comparan los datos reales con las predicciones, así como a los intervalos de predicción.

4.3.3 Descripción del modelo SARIMAX y resultados obtenidos

Los hiperparámetros asociados a SARIMAX comprenden los componentes de autoregresión (AR), integración (I) y media móvil (MA), tanto para la serie completa como para la componente estacional (S).

Para comprobar la estacionariedad de los datos, se ha realizado la prueba ADF (Augmented Dickey-Fuller [25]) la cual es una prueba estadística utilizada para determinar si una serie temporal es estacionaria o no. Dicha prueba se basa en el modelo autorregresivo (AR) y permite analizar la presencia de una raíz unitaria en la serie. Es un análisis para describir la persistencia o la falta de estacionariedad en los datos, es decir, analizar si los efectos pasados en la serie temporal tienen un impacto duradero o se desvanecen a medida que avanza el tiempo.

En nuestro caso, tras realizar dicha prueba, nos encontramos con unos datos que no presentan estacionariedad.

El modelo se ha ajustado manteniendo el período de variación estacional fijo en 32, que corresponde al número de servicios diarios realizados por la línea de transporte. Se han explorado todas las combinaciones posibles de hiperparámetros, probando valores de 0 a 5, excepto para el período estacional. La selección de la mejor combinación se realizó mediante el criterio de información Akaike (AIC). El AIC es una métrica que evalúa la calidad del ajuste del modelo a los datos de entrenamiento, teniendo en cuenta la verosimilitud, y penaliza la complejidad del modelo al considerar el número de parámetros. Esta penalización permite evitar el sobreajuste de modelos complejos a los datos.

En la figura 4.11 se presenta una comparación entre los valores reales y las predicciones generadas por el modelo con el mejor ajuste obtenido. Este resultado proporcionó los siguientes valores para las métricas de evaluación:

- Error absoluto medio (MAE) = 9.3
- Raíz cuadrada del error cuadrático medio (RMSE) = 12.7

Considerando que la cantidad media de pasajeros que manejan los datos, es de 37, podemos concluir que la precisión del modelo es alta.

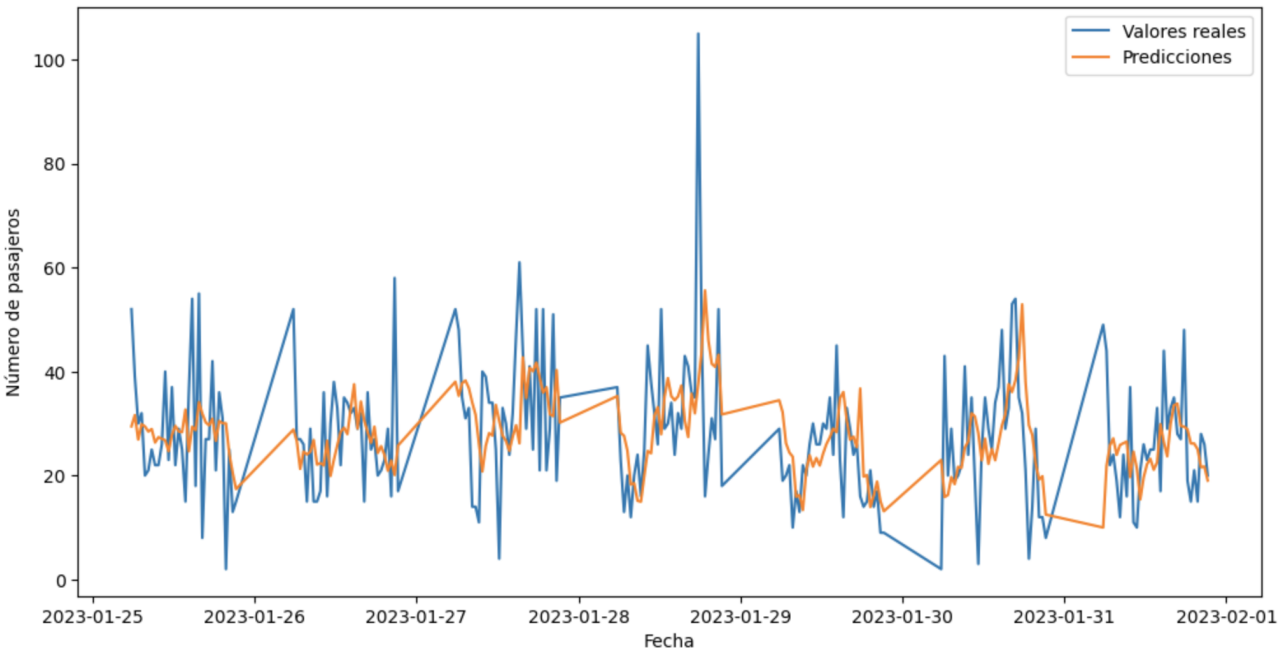

Figura 4.11: Resultado SARIMAX (5, 1, 0) x (2, 0, 0, 32)

Con el propósito de realizar una verificación adicional del modelo predictivo, se han llevado a cabo una serie de diagnósticos sobre el modelo entrenado. En la figura 4.12, se presentan los resultados correspondientes al residuo estandarizado (Standardized Residual), en el cual no se evidencia la presencia de patrones claros, pero se observa una consistencia relativa a lo largo del gráfico. Además, se presentan los resultados para el histograma junto con la estimación de la densidad (Histogram plus estimated density), en donde la curva de densidad estimada (KDE) se asemeja en gran medida a la distribución normal (N(0, 1)), aunque se observa una elevación.

En relación a esto, en la figura 4.13 se presentan los resultados obtenidos del gráfico de probabilidad normal Q-Q (Normal Q-Q). Se observa que la gran mayoría de los datos se encuentran en línea con la distribución normal esperada, con excepción de una desviación que se aprecia al final de la línea. Además, se muestran los resultados del correlograma, donde se puede observar que la mayoría de los datos se encuentran dentro de la zona grisácea, que representa nuestro intervalo de confianza. Esta información adicional proporciona indicios de que el modelo se ajusta adecuadamente a los supuestos y que los residuos o errores del modelo presentan una estructura aleatoria dentro de los límites de confianza establecidos.

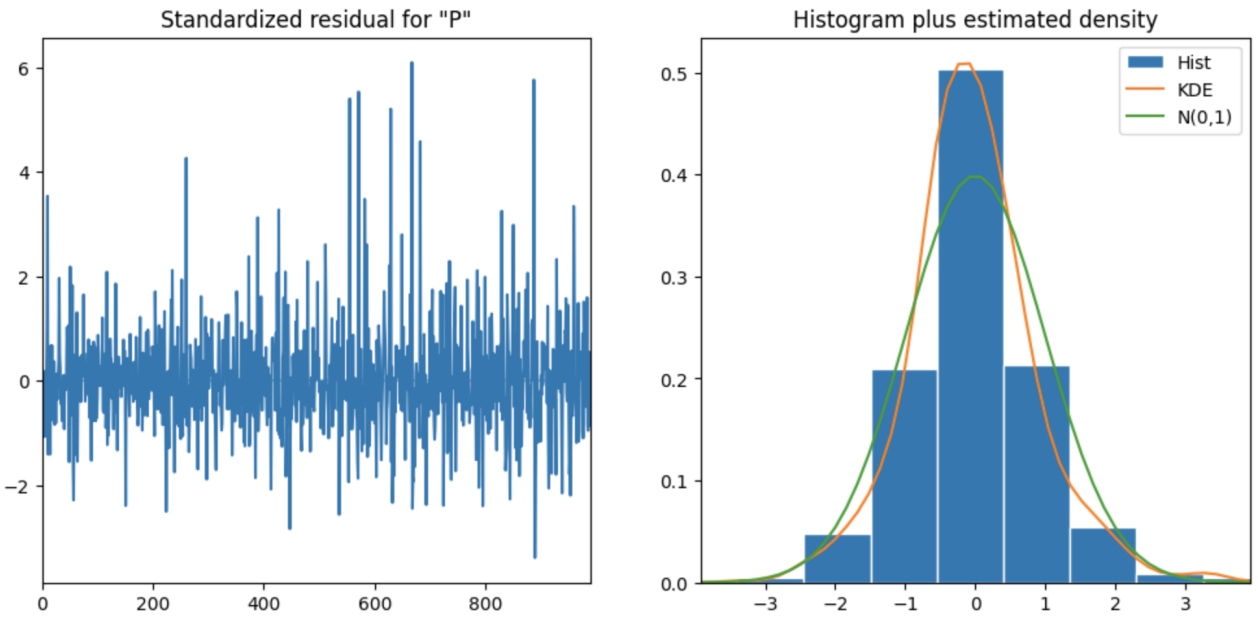

Figura 4.12: Plots validaciones (1)

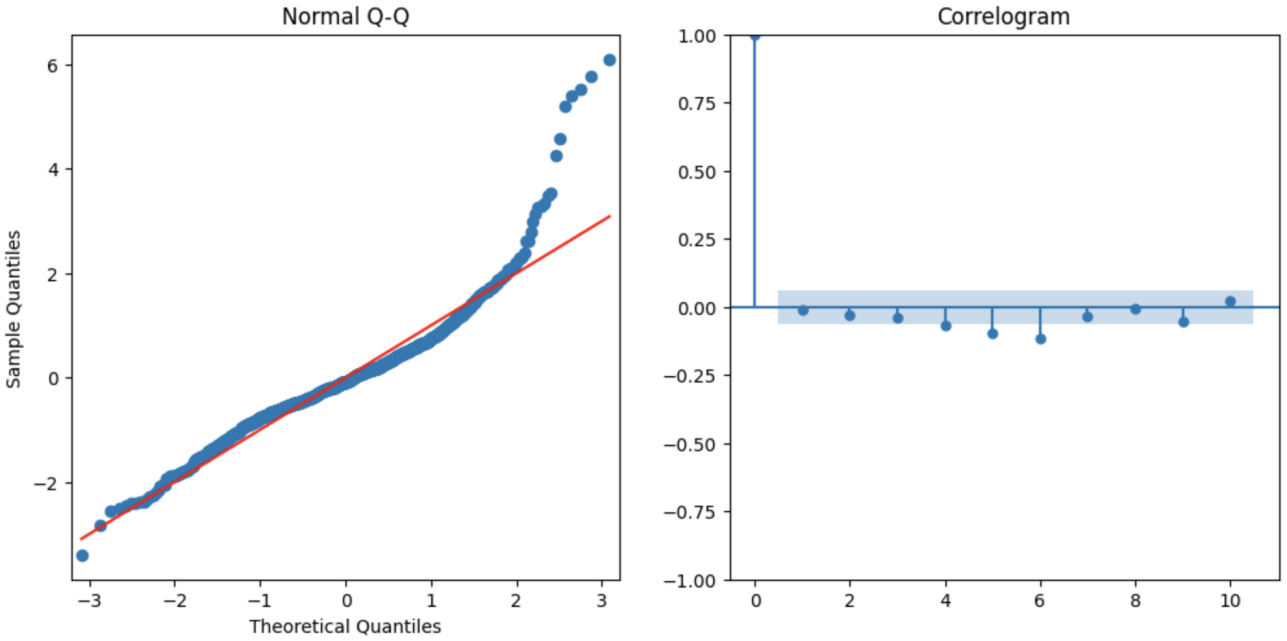

Figura 4.13: Plots validaciones (2)

4.3.4 Descripción del modelo Neural Prophet y resultados obtenidos

La cantidad de hiperparámetros asociados a Neural Prophet es considerablemente mayor en comparación con SARIMAX. Es por ello que no se llevó a cabo un ajuste tan exhaustivo como en el caso de SARIMAX. En su lugar se aprovecharon los mejores resultados obtenidos con SARIMAX y se probaron los mismos parámetros con Neural Prophet. Esta estrategia se adoptó para simplificar el proceso de ajuste y comparar el rendimiento relativo entre ambos modelos.

No obstante, se manipularon otros hiperparámetros como el modo de estacionalidad, la cantidad de epochs, el batch size, entre otros.

El modelo Neural Prophet, también se ha ajustado manteniendo el período estacional fijo en 32 y se han explorado las mejores combinaciones obtenidas por el criterio Akaike realizado anteriormente. En la figura 4.14 se presenta una comparación entre los valores reales y las predicciones generadas por el modelo con el mejor ajuste obtenido. Este resultado proporcionó los siguientes valores para las métricas de evaluación:

- Error absoluto medio (MAE) = 6.0
- Raíz cuadrada del error cuadrático medio (RMSE) = 8.6

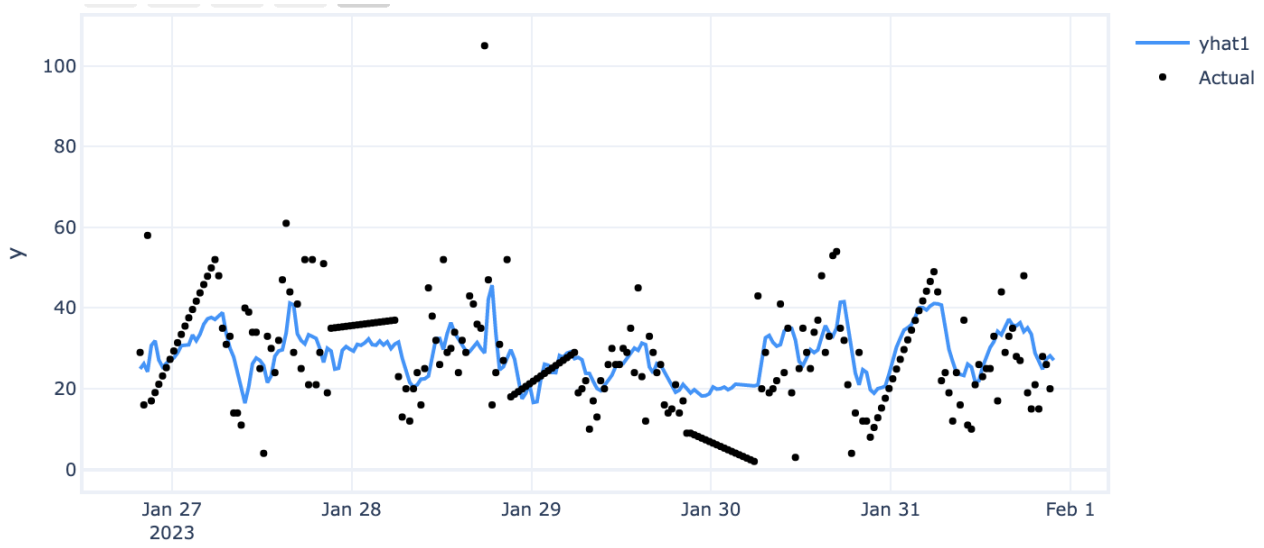

Figura 4.14: Resultado Neural Prophet (5, 1, 0) x (2, 0, 0, 32)

Comparando el resultado de Neural Prophet con el de SARIMAX, podemos notar una mejoría en la precisión y el rendimiento del modelo. También se debe tener en cuenta el coste

computacional que tiene Neural Prophet al hacer uso de Redes Neuronales LSTM.

Para visualizar en profundidad el comportamiento de Neural Prophet, podemos estudiar su tendencia o su estacionalidad semanal. En las figuras 4.15 y 4.16 se puede observar como se define una estacionalidad semanal, donde la cantidad de pasajeros los fines de semana es inferior a los días entre semana. Además podemos observar como varía la tendencia a lo largo del mes de enero.

Figura 4.15: Tendencia y estacionalidad

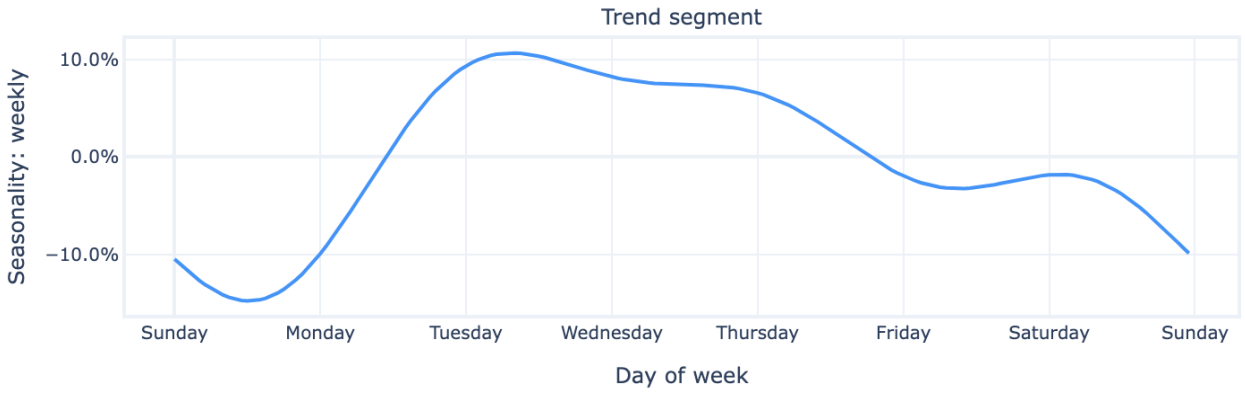

Figura 4.16: Estacionalidad semanal

Durante el transcurso del proyecto, se han explorado otros modelos como ARIMA y redes

neuronales con dependencia del contexto, como LSTM. Sin embargo, los resultados obtenidos no han sido lo suficientemente satisfactorios como para considerarlos. En el caso del modelo ARIMA, se observó que su capacidad de ajuste era limitada, como se aprecia en la figura 4.17. Por otro lado, se descartó el uso de redes neuronales, como LSTM, debido a la escasez de datos disponibles para realizar un entrenamiento óptimo y obtener resultados aceptables.

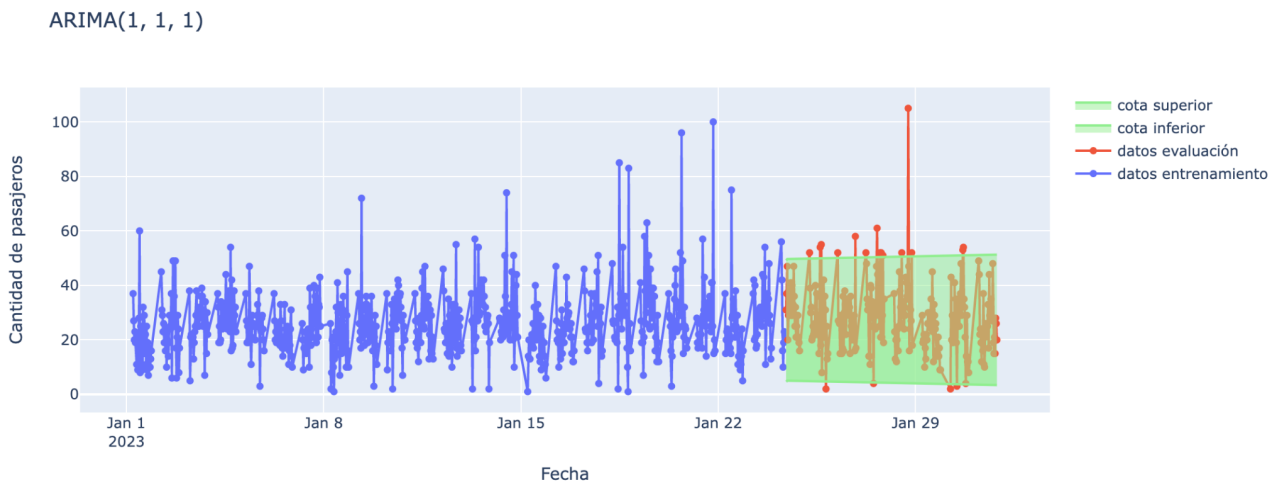

Figura 4.17: Resultados ARIMA (1, 1, 1)

Capítulo 5 Conclusiones y líneas futuras

5.1 Conclusiones

En este proyecto, se han desarrollado dos líneas de trabajo independientes que comparten el objetivo de mejorar la transparencia de la empresa con los usuarios, proporcionar acceso rápido a los datos sin requerir conocimientos técnicos y anticipar posibles situaciones de saturación a través de la predicción de datos.

En la primera parte del proyecto, se ha logrado exitosamente la creación de una herramienta de consulta de datos históricos mediante una interfaz basada en Telegram. Esta herramienta, que utiliza un chatbot personalizado, ha demostrado ser una solución efectiva para proporcionar respuestas rápidas a los usuarios. La integración de la tecnología GPT ha mejorado significativamente la experiencia del chatbot al ofrecer respuestas coherentes y relevantes.

Por otro lado, en la fase de predicción de pasajeros, aunque se ha realizado un análisis exhaustivo y exploración de los datos (se destaca que solo hemos trabajado con un mes de datos), se han aplicado diversas técnicas de preprocesamiento y transformación, identificando patrones y tendencias relevantes en las series temporales analizadas. Se han implementado varios modelos, como ARIMA, SARIMAX y Neural Prophet, con el objetivo de encontrar el modelo más adecuado para nuestros datos.

Entre estos modelos, Neural Prophet ha mostrado resultados prometedores. Sin embargo, se ha observado que este modelo requiere una capacidad computacional significativa. Por otro lado, SARIMAX ha ofrecido resultados similares y su costo computacional ha sido más manejable debido a la cantidad limitada de datos disponibles. En definitiva, el uso de varios modelos de manera simultánea ha proporcionado enfoques complementarios que resultan valiosos para la toma de decisiones.

En conclusión, este proyecto ha logrado avanzar en dos líneas de trabajo paralelas, centradas en mejorar la transparencia empresarial, facilitar el acceso a datos y anticipar situaciones de saturación. La creación de una herramienta de consulta basada en Telegram ha mejorado la interacción con los usuarios, mientras que la implementación de modelos de predicción ha permitido obtener insights valiosos. Estos resultados respaldan la importancia de seguir explorando y mejorando estas áreas de trabajo para ofrecer soluciones eficaces y beneficiosas para la empresa y sus usuarios.

5.2 Líneas futuras

Hay varias líneas por las que se puede seguir el proyecto, en ambas partes del mismo. A continuación se enumeran unas posibilidades:

- Mejora de la integración con GPT haciendo uso del guardado del historial del chat.

- Añadir funcionalidades nuevas, no solo la consulta a la cantidad de pasajeros, sino también escalar a diversas áreas de la empresa.

- Realizar la serie temporal de cada una de las paradas de una línea de transporte, retroalimentando éstas con la predicción en $t - 1$.

- Hacer uso de nuevos conjuntos de datos que sean relevantes para la predicción, como el tiempo meteorológico.

Capítulo 6 Summary and Conclusions

6.1 Conclusions

In this project, two independent lines of work have been developed with the shared objective of enhancing transparency between the company and its users, providing quick access to data without requiring technical knowledge, and anticipating potential saturation situations through data prediction.

In the first part of the project, the successful development of a historical data query tool through a Telegram-based interface has been achieved. This tool, utilizing a customized chatbot, has proven to be an effective solution in delivering prompt responses to users. The integration of GPT technology has significantly enhanced the chatbot experience by providing coherent and relevant responses.

On the other hand, in the passenger prediction phase, a comprehensive analysis and exploration of the data has been conducted. Various preprocessing and transformation techniques have been employed to identify relevant patterns and trends within the analyzed time series. Several models, including ARIMA, SARIMAX, and Neural Prophet, have been implemented to determine the most suitable model for the data.

Among these models, Neural Prophet has shown promising results. However, it has been observed that this model requires significant computational capacity. On the other hand, SARIMAX has provided similar results with more manageable computational costs due to the limited amount of available data. Ultimately, the use of multiple models concurrently has provided complementary approaches that prove valuable in decision-making processes.

In conclusion, this project has advanced two parallel lines of work focused on enhancing corporate transparency, facilitating data access, and anticipating saturation situations. The development of a Telegram-based query tool has improved user interaction, while the implementation of prediction models has yielded valuable insights. These results underscore the importance of continued exploration and improvement in these areas of work to offer effective and beneficial solutions for the company and its users.

6.2 Future Work

There are several avenues to explore for further development of the project, in both parts. Here are some possibilities:

- Enhancing integration with GPT by utilizing chat history storage.

- Adding new functionalities beyond passenger quantity queries. Expanding the scope of the tool to encompass various areas of the company would provide users with more comprehensive and versatile experience.

- Creating time series analysis for each stop on a transportation line.

- Incorporate relevant external data, such as weather conditions.

Capítulo 7 Presupuesto

7.1 Costes hardware

Tipos	Descripción	Coste
Macbook Pro	Portátil desde el que se realizó la primera parte del proyecto	196,87 €
Ordenador sobremesa por piezas	Ordenador desde el que se entrenaron los modelos y se evaluaron	291,66 €

Tabla 7.1: Costes hardware

7.2 Costes Recursos Humanos

Horas de trabajo	Coste/Hora	Coste total
280h	30€/h	8.400,00 €

Tabla 7.2: Costes Recursos Humanos

7.3 Costes totales del proyecto

Coste Hardware	Coste RRHH	Coste proyecto
488,53 €	8.400,00 €	8.888,53 €

Tabla 7.3: Costes totales del proyecto

Bibliografía

1. TITSA. (s.f.). Transportes Interurbanos de Tenerife S.A. Recuperado el 22 de mayo de 2023, de https://titsa.com/
2. Janssen, M., Charalabidis, Y., & Zuiderwijk, A. (2012). Benefits, adoption barriers and myths of open data and open government. Information Systems Management, 29(4), 258-268. DOI: 10.1080/10580530.2012.716740 de https://journals.sagepub.com/doi/10.1111/j.1467-9248.2011.00915.x
3. Consejería de Administraciones Públicas, Justicia y Seguridad del Gobierno de Canarias. (s.f.). Datos abiertos del Gobierno de Canarias. Recuperado el 22 de mayo de 2023, de http://datos.canarias.es/
4. TeamGantt. (s.f.). Project Management Software & Team Collaboration Tools. Recuperado el 22 de mayo de 2023, de https://www.teamgantt.com/
5. Goldbloom, A. (2018, 16 de agosto). Machine Learning (ML) vs Artificial Intelligence (AI): Crucial Differences. Data Science Central. Recuperado el 22 de mayo de 2023, de https://www.datasciencecentral.com/machine-learning-ml-vs-artificial-intelligence-ai-crucial-differences/
6. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444. DOI: https://doi.org/10.1038/nature14539
7. Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, 65(6), 386-408. Dirección: https://psycnet.apa.org/doiLanding?doi=10.1037%2Fh0042519
8. Hinton, G. (2010). A practical guide to training restricted Boltzmann machines. Neural Networks: Tricks of the Trade, 2, 599-619. DOI: 10.1007/978-3-642-35289-8_36
9. Brownlee, J. (s.f.). Rectified Linear Activation Function for Deep Learning Neural Networks. Machine Learning Mastery. Recuperado el 22 de mayo de 2023, de https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/
10. Verma, A. (2019, 22 de abril). Stochastic Gradient Descent Clearly Explained. Towards Data Science. Recuperado el 22 de mayo de 2023, de https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31
11. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324. DOI: 10.1109/5.726791
12. Uber. (s.f.). Forecasting: An Introduction. Recuperado el 22 de mayo de 2023, de https://www.uber.com/en-ES/blog/forecasting-introduction/
13. Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: Principles and Practice (2nd ed.). OTexts. Recuperado el 22 de mayo de 2023, de https://otexts.com/fpp2/
14. Brownlee, J. (s.f.). How to Check if Time Series Data is Stationary with Python. Machine Learning Mastery. Recuperado el 22 de mayo de 2023, de https://machinelearningmastery.com/time-series-data-stationary-python/

15. Christopher Lewis. (2021). Forecasting using Facebook's Prophet library. Analytics Vidhya. Dirección: https://medium.com/analytics-vidhya/forecasting-using-facebooks-prophet-library-ce628e76586b
16. Valentin Catherine. (2022). Forecasting using Facebook's Prophet library. Analytics Vidhya. Dirección: https://towardsdatascience.com/in-depth-understanding-of-neuralprophet-through-a-complete-example-2474f675bc96
17. Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013). Bayesian data analysis (3rd ed.). CRC Press.
18. Radford, A., & GPT-3 Team. (2020). Learning to summarize with human feedback. OpenAI Blog. Recuperado el 22 de mayo de 2023, de https://openai.com/blog/learning-to-summarize-with-human-feedback/
19. OpenAI. (s.f.). OpenAI. Recuperado el 22 de mayo de 2023, de https://openai.com/
20. Khurana, D., Koli, A., Khatter, K. et al. Natural language processing: state of the art, current trends and challenges. Multimed Tools Appl 82, 3713–3744 (2023). https://doi.org/10.1007/s11042-022-13428-4
21. Telegram. (s.f.). Telegram Bot API. Recuperado el 22 de mayo de 2023, de https://core.telegram.org/bots/api
22. Microsoft. (s.f.). SQL Server. Recuperado el 22 de mayo de 2023, de https://www.microsoft.com/en-us/sql-server/
23. GitHub. (s.f.). GitHub. Recuperado el 22 de mayo de 2023, de https://github.com/
24. Telegram. (s.f.). BotFather. Recuperado el 22 de mayo de 2023, de https://core.telegram.org/bots#botfather
25. Dickey, D. A., & Fuller, W. A. (1979). Distribution of the estimators for autoregressive time series with a unit root. Journal of the American Statistical Association, 74(366a), 427-431. DOI: https://doi.org/10.1080/01621459.1979.10482531