Prueba Técnica para Ingeniero en IA Generativa

# Parte 1: Evaluación Teórica

## 1.1. Conceptos Estadísticos y Análisis Predictivo

**Pregunta:** Explica cómo utilizarías modelos de regresión para predecir tendencias en un conjunto de datos de series temporales. Proporciona un ejemplo práctico.

Para analizar series de tiempo se deben tener en cuenta ciertos factores tales como la estacionalidad, periodicidad y uniformidad, dependiendo de eso se pueden elegir varias opciones de modelos predictivos clásicos como modelos ARIMA, o se pueden aprovechar técnicas de machine learning ( eg XGboost) o bien las técnicas de deep learning (LSTM) también son una buena opción.

Usualmente mi enfoque principal es convertir esto en un problema de aprendizaje supervisado usando la misma salida de la serie de tiempo como entradas pasadas para predecir un valor futuro.

Por ejemplo

Si se tiene la siguiente serie 1,2,3,4,5,6,7,8 los datos se deben organizar dependiendo de un tamaño de ventana escogido previamente, ejemplo:

m=3

Los datos quedarían de la siguiente manera

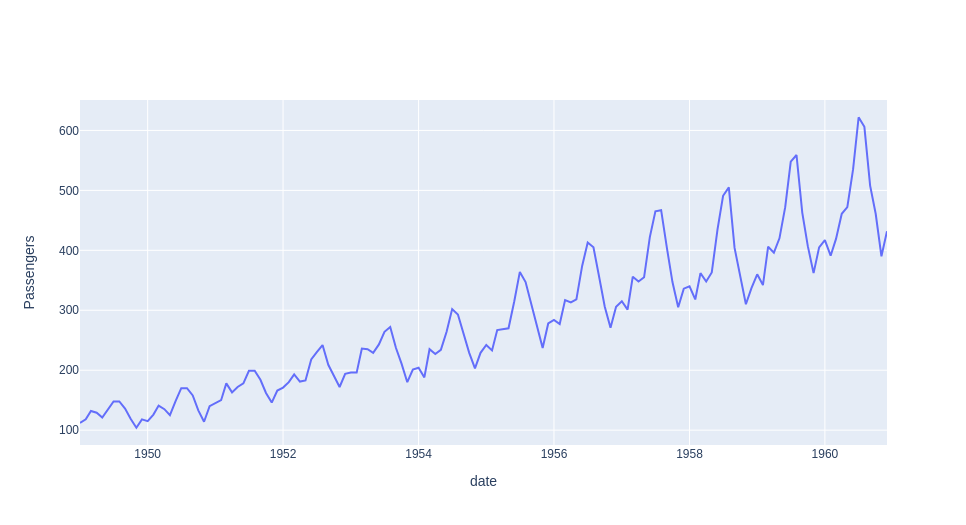
|  |  |  |  |
| --- | --- | --- | --- |
| **X1** | **X2** | **X3** | **y** |
| 1 | 2 | 3 | 4 |
| 2 | 3 | 4 | 5 |
| 3 | 4 | 5 | 6 |
| 4 | 5 | 6 | 7 |
| 5 | 6 | 7 | 8 |

De esta manera ya tenemos un conjunto de características y un conjunto salida que puede ser entrenado con un algoritmo de regresión clásico o con una red neuronal preferiblemente recurrente abordando todo el proceso habitual de machine learning que involucra el preprocesamiento, entrenamiento y validación del modelo con un error que podría ser el MSE, RMSE, entre otros.

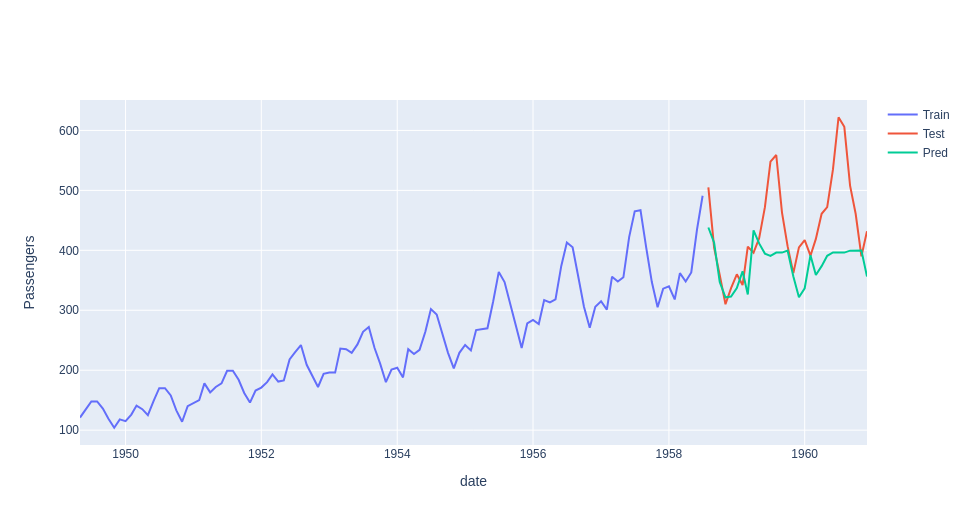
Ejemplo práctico:

EL código de este ejemplo se encuentra en **Parte1/evaluacion teorica/xgboost\_forecasting.ipynb**

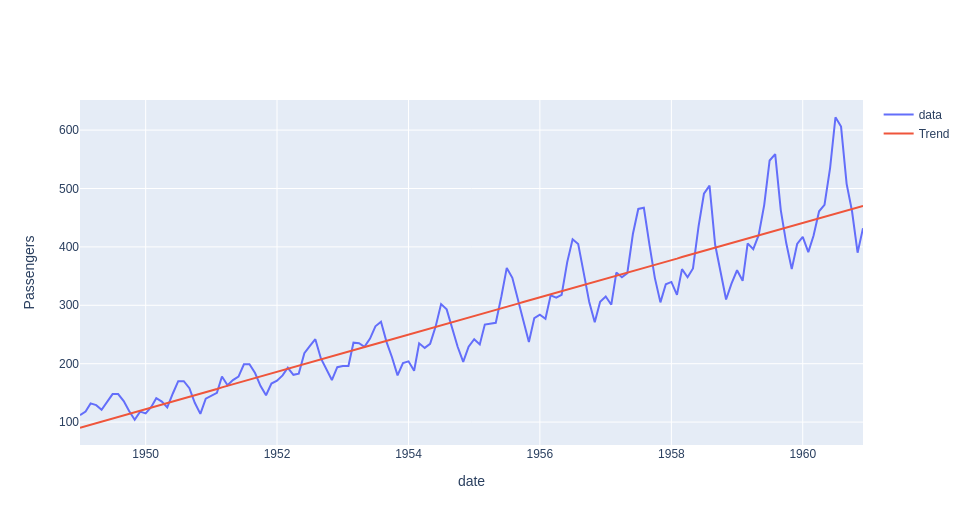
Utilizaré un dataset famoso ("<https://raw.githubusercontent.com/jbrownlee/Datasets/master/airline-passengers.csv>) que muestra la cantidad de pasajeros en una fecha determinada en una aerolínea. EL conjunto de datos se ve de la siguiente manera



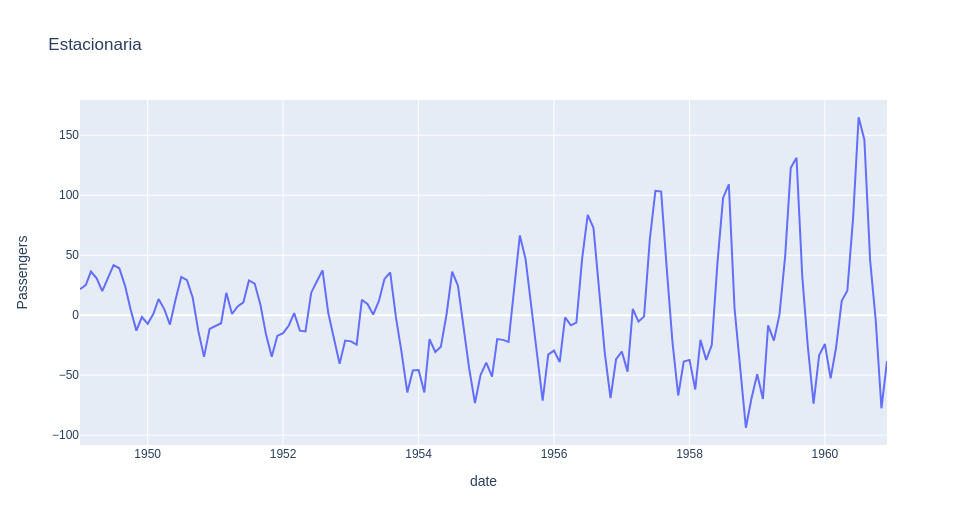
Podemos observar que esta serie no tiene u promedio fijo en el tiempo sino que este va aumentando, es decir no es una serie estacionaria. Si entrenamos el modelo de este ejemplo con una serie de este tipo tenemos una salida como esta



Vemos que el modelo no genera predicciones adecuadas y esto se debe a la no estacionalidad de la serie. En ese caso, hay que hacer una transformación primero tratando de detectar como es la tendencia de la serie, en este caso se puede observar que la serie tiene un tipo de estacionalidad lineal por lo que a través de una regresión líneal podemos estimarla

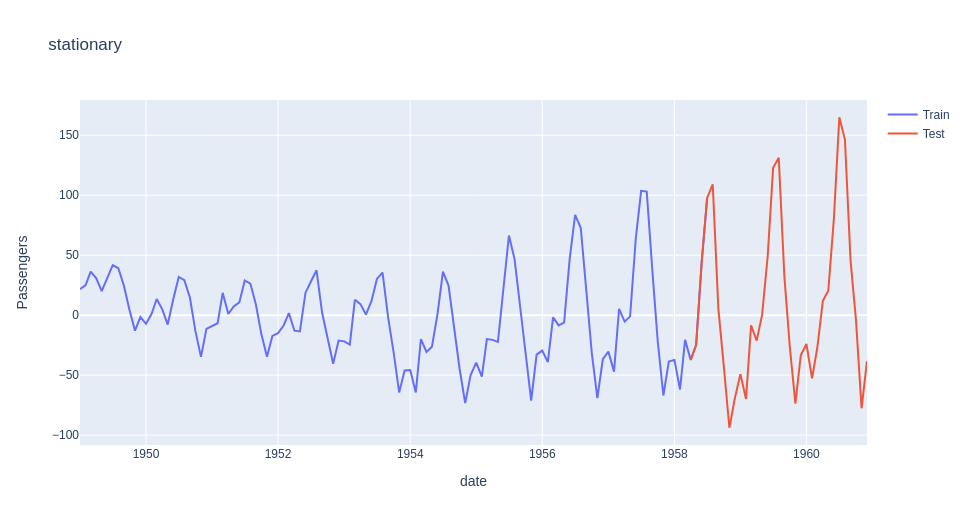


Lo siguiente es restar esa tendencia de la serie original para obtener la siguiente serie:



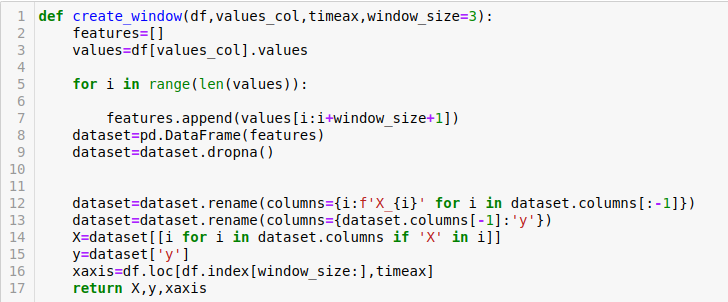
A pesar de no estar perfectamente estacionaria si vemos un promedio un poco más constante

Ahora bien una vez tenemos esta serie el siguiente paso es crear nuestro conjunto de entrenamiento y prueba, se escoge un 80% para entrenamiento y un 20% para prueba

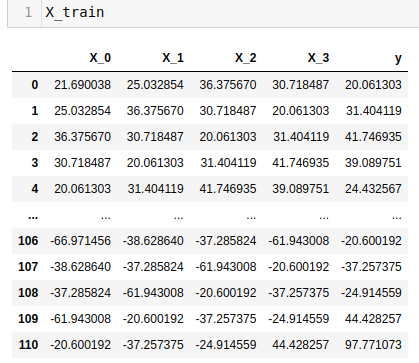
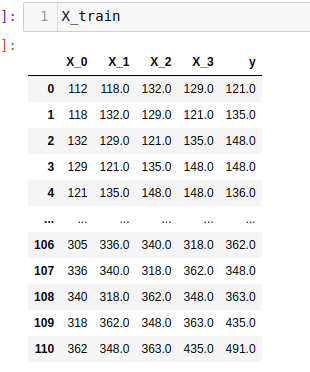


Una vez tenemos estos datos, lo importante es transformarlos de tal manera que podamos modelar a través de aprendizaje supervisado, se escogerá el modelo XGBoostRegressor para la tarea.

A través de la siguiente función se transformarán los datos a través de una ventana como se explicó al inicio de este análisis

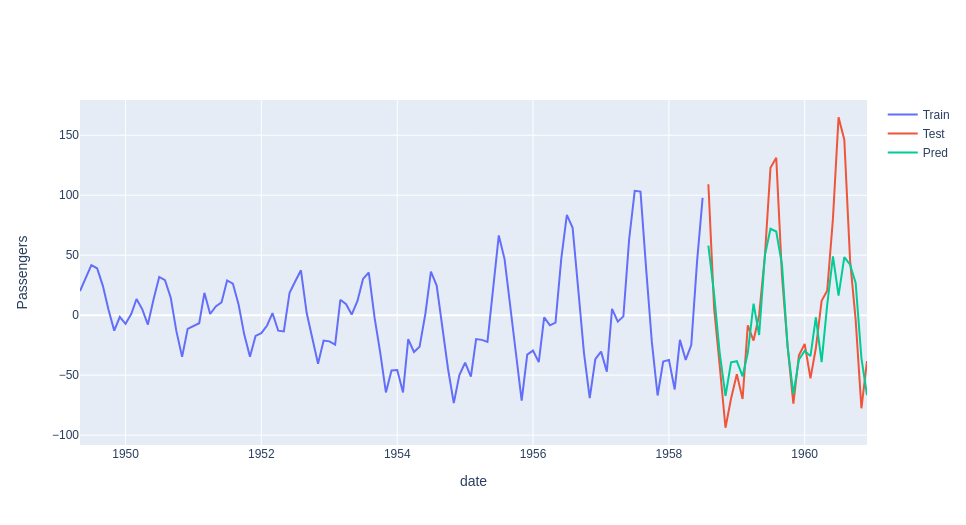


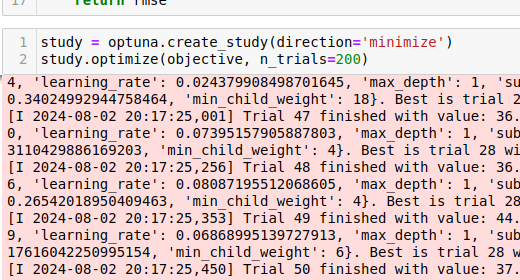
Si escogemos un valor de ventana de valor 4 tendremos el siguiente dataset ( eg entrenamiento)



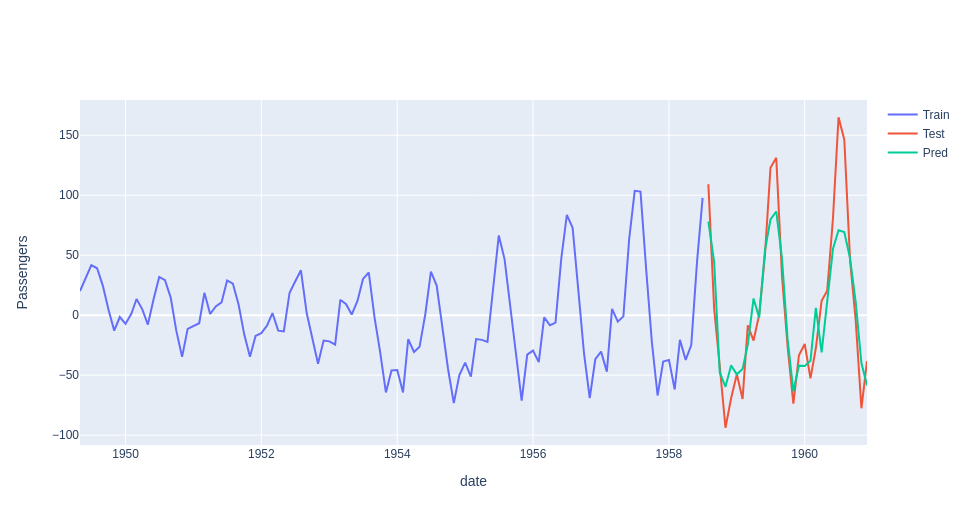
Datos con tendencia datos sin tendencia

De esta manera ya podemos entrenar nuestro XgBoost como si fuera un modelo de regresión simple, El resultado es el siguiente:

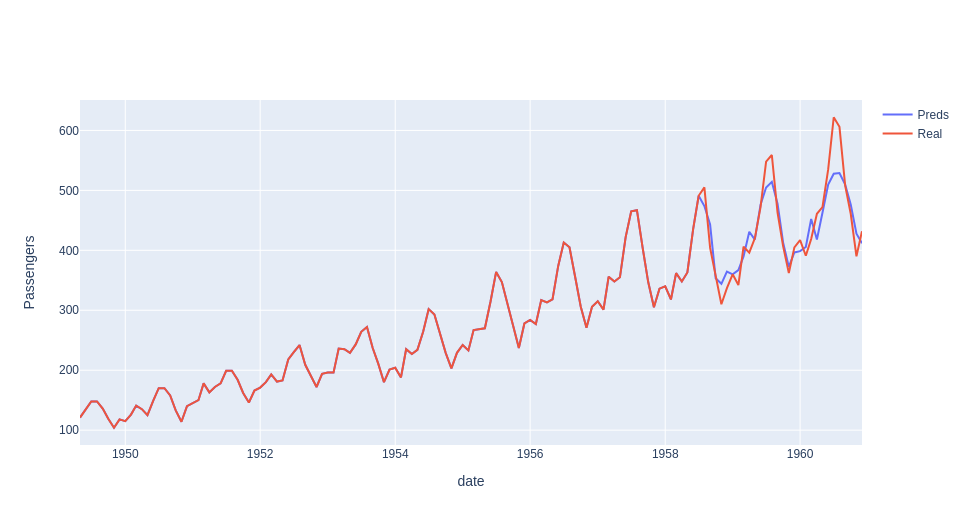
Este modelo nos deja un RMSE: 91.43 por lo tanto, es posible que tengamos que hacer un afinamiento de hiperparámetros ( usaremos la biblioteca optuna)



Al finalizar los 200 experimentos, escogimos aquellos hiperparámetros que generaron el menor RMSE de todos los experimentos, el resultado es el siguiente:

EL RMSE arrojado en este caso es de 33.2 lo cual mejora nuestro modelo inicial

Finalmente se agrega la tendencia de nuevo para tener nuestra gráfica original y de esta manera tener un mejor modelo respecto al inicial



## 1.2. Modelos de Lenguaje de Gran Escala

**Pregunta:** Describe cómo utilizarías BERT para una tarea de clasificación de textos. Incluye los pasos desde la pre-procesamiento de datos hasta la evaluación del modelo.

Para establecer un modelo Bert en una tarea de clasificación de textos se deben tener en cuenta 3 pipelines:

1. Pipeline de características: Donde se hace la extracción y preprocesamiento de los documentos, aquí revisamos que tipo de datos tenemos o si tenemos clases desbalanceadas, en ese caso es necesario en lo posible conseguir más información o generar textos sintéticos con ayuda de otros modelos. Como estamos hablando de un modelo de clasificación de texto, es posible que haya que hacer tareas de stemming, lematización, remoción de stopwords (aunque a veces eso puede cambiar el contexto en el transformer y puede crear mal desempeño), signos de puntuación entre otros.
2. Pipeline de entrenamiento: Se define el modelo a utilizar y se realiza el posterior entrenamiento en este caso Bert.

El tokenizer juega un papel importante ya que de aquí obtenemos los tokens ids y las attention masks, además configuramos ciertos parámetros relacionados con la longitud máxima de nuestros textos para que sean truncados

1. Pipeline de evaluación : Como se tiene un problema de clasificación se pueden definir métricas de Accuracy, recall, precision ,F1 score, Curva Roc y Ks plot, dependiendo de estas métricas es posible que haya que volver al pipeline de entrenamiento ya sea para afinar hiperparámetros o conseguir más información por temas de overfitting o hacer el modelo mas complejo si se ven eventos de underfitting.

# Parte 2: Evaluación Práctica

## 2.1. Implementación de Modelos de IA Generativa

**Tarea:** Implementa un modelo generativo utilizando Mistral o un modelo de OpenAI para generar texto basado en un conjunto de datos proporcionado.

**Requisitos:** Utiliza PyTorch o TensorFlow, y demuestra cómo entrenar el modelo, generar nuevas secuencias de texto, y evaluar la calidad del texto generado.

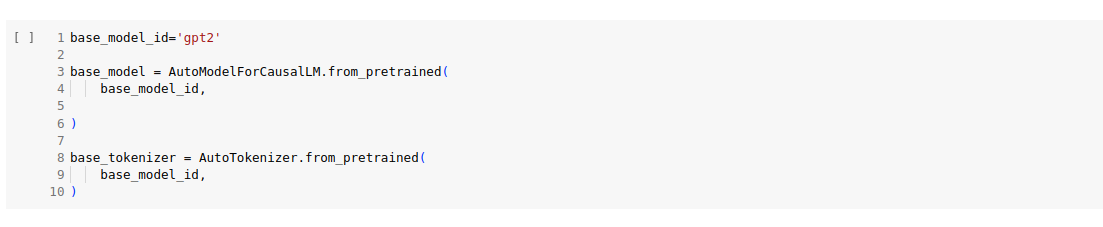
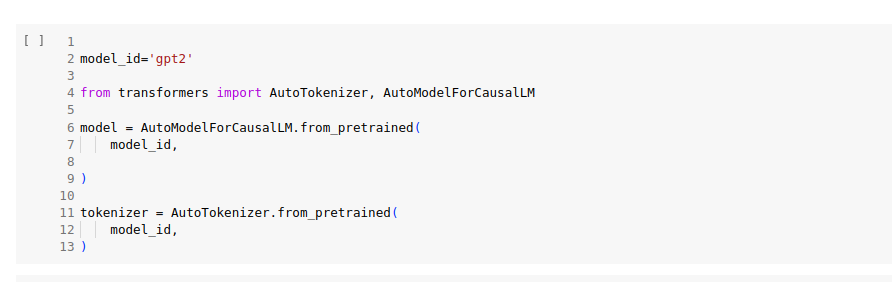
**Solución:**

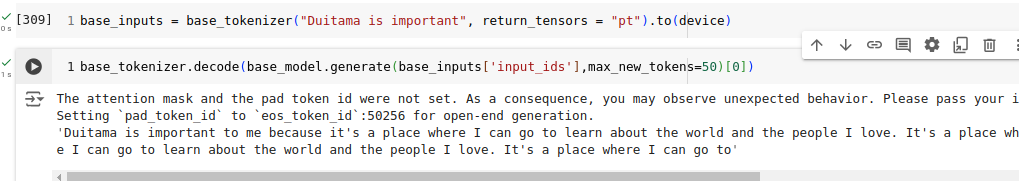
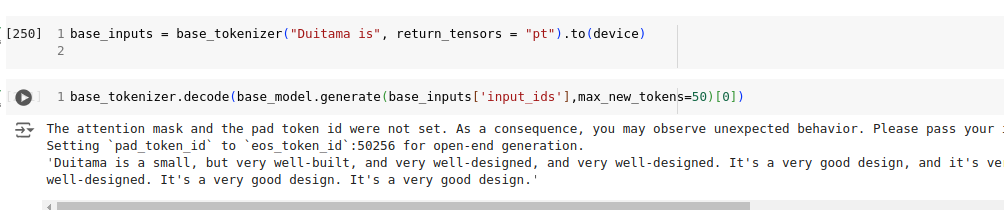
En este caso escogí entrenar el modelo de OpenAI GPT2 especialmente por su bajo consumo computacional respecto a otros modelos teniendo en cuenta que solo cuento en el momento con google colab gratuito. Para este caso decidí crear un generador de texto usando como fuente de datos La historia de Duitama que aparece en la página de Wikipedia. Por otro lado,el archivo para hacer el entrenamiento llama **duitama.es.en.txt,** el entrenamiento se encuentra en el notebook **GPT2 fine tunning.ipynb**

Opté por usar la libreria transformers y pytorch para hacer un entrenamiento que me permitiera enfocarme mas en el desarrollo del modelo y no en el código como tal.

Pasos desarrollados

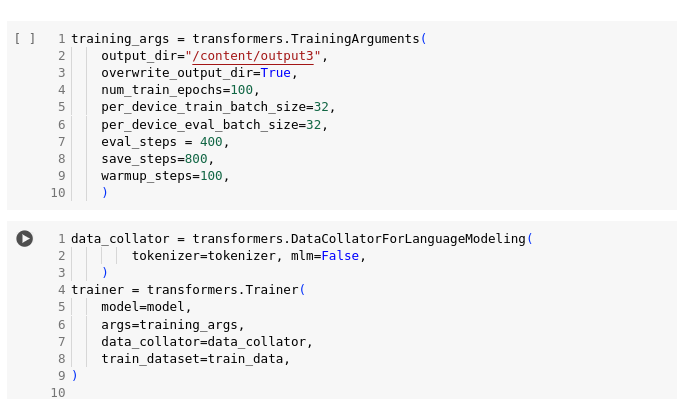
1. Crear el dataset a través del archivo de texto, esto se logró con esta instrucción 
2. Antes de utilizar el dataset instancié dos modelos, uno de línea base y otro con el que voy a hacer el finetunning, Para probar el modelo de linea base hice un par de prompts sobre Duitama y estas fueron las respuestas

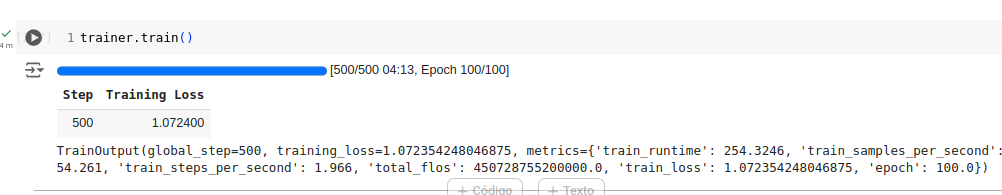
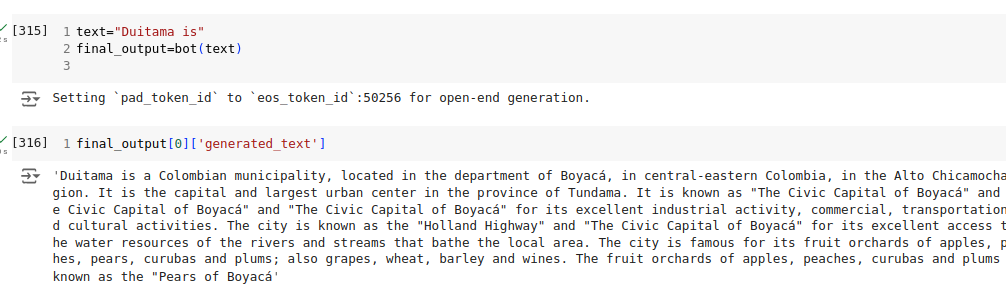


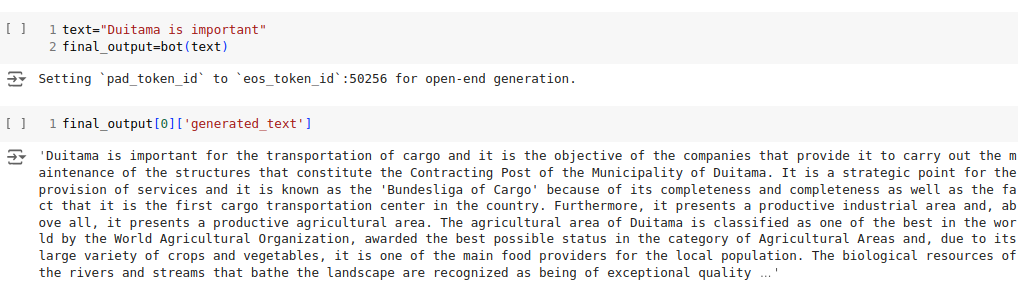


Como se puede evidenciar , las respuestas dadas por GPT2 no son muy precisas, además el valor de las respectivas perplexidades estaban en valores superiores a 1000.

1. En ese caso ahora si se hace el respectivo finetunning con el archivo de texto mencionado definiendo el respectivo trainer y algunos hiperparámetros. El modelo se entrenó por 100 épocas con los siguientes resultados:



1. 
2. Una vez finalizado el entrenamiento se procede a crear un pipeline de inferencia con los mismos prompts iniciales

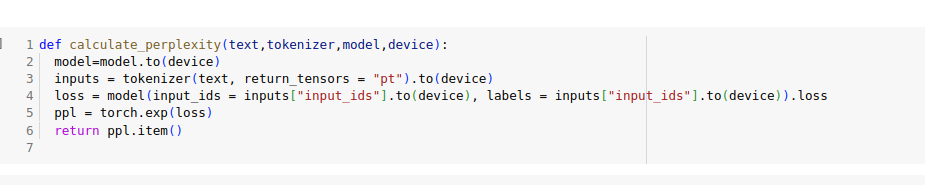


En conclusión las respuestas del modelo se ajustan mucho más a la realidad y por ende las perplexidades de las salidas disminuyeron drásticamente como se muestra en la siguiente tabla:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Modelo | Prompt 1 “Duitama is” | Prompt 2 “Duitama is important” | Perplexity (Prompt1) | Perplexity (Prompt2) |
| **GPT2 Base Line** | Duitama is a small, but very well-built, and very well-designed, and very well-designed. It's a very good design, and it's very well-designed. It's a very good design. It's a very good design. It's a very good design. It's a very good design. It's a very good design. It's a very good design. It's a very good design. It's a very good design. It's a very good design. It's a very good design. It's a very good design. It's a very good design. It's a very good design | Duitama is important to me because it's a place where I can go to learn about the world and the people I love. It's a place where I can go to learn about the world and the people I love. It's a place where I can go to learn about the world and the people I love. I'm a big fan of the show, and I'm really excited to see what the next season will bring. I'm a big fan of the show, and I'm really excited to see what the next season will bring. | 1399.7 | 1598.38 |
| **GPT2 Trained** | Duitama is a Colombian municipality, located in the department of Boyacá, in central-eastern Colombia, in the Alto Chicamocha region. It is the capital and largest urban center in the province of Tundama. It is known as "The Civic Capital of Boyacá" and "The Civic Capital of Boyacá" and "The Civic Capital of Boyacá" for its excellent industrial activity, commercial, transportation and cultural activities. The city is known as the "Holland Highway" and "The Civic Capital of Boyacá" for its excellent access to the water resources of the rivers and streams that bathe the local area. The city is famous for its fruit orchards of apples, peaches, pears, curubas and plums; also grapes, wheat, barley and wines. The fruit orchards of apples, peaches, curubas and plums are known as the "Pears of Boyacá | Duitama is important for the transportation of cargo and it is the objective of the companies that provide it to carry out the maintenance of the structures that constitute the Contracting Post of the Municipality of Duitama. It is a strategic point for the provision of services and it is known as the 'Bundesliga of Cargo' because of its completeness and completeness as well as the fact that it is the first cargo transportation center in the country. Furthermore, it presents a productive industrial area and, above all, it presents a productive agricultural area. The agricultural area of Duitama is classified as one of the best in the world by the World Agricultural Organization, awarded the best possible status in the category of Agricultural Areas and, due to its large variety of crops and vegetables, it is one of the main food providers for the local population. The biological resources of the rivers and streams that bathe the landscape are recognized as being of exceptional quality and, owing to their high quality, | 1.08 | 70.11 |

Se podrían organizar más conjuntos de prompts para evaluar cómo se comporta la perplexidad del modelo y de esa manera decidir si se necesita más información o hacer un ajuste en hiperparámetros.

La función para calcular la perplexidad se codificó de la siguiente manera:



## 2.2. Aplicación de Aprendizaje por Refuerzo

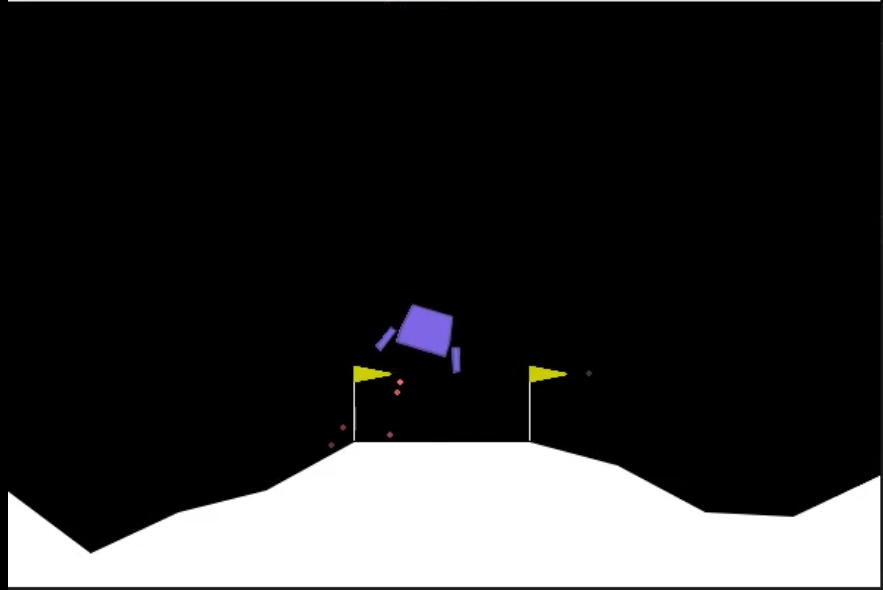
**Tarea:** Desarrolla un algoritmo de aprendizaje por refuerzo para optimizar una política en un entorno de simulación (por ejemplo, OpenAI Gym).

**Requisitos:** Explica el enfoque utilizado, implementa el algoritmo y muestra los resultados obtenidos.

**Solución:**

La solución para este problema se encuentra en el notebook **Lunarlander.ipynb desarrollado en pytorch**

Se escogió el ambiente Lunar Lander de Gym para desarrollar este problema debido a que es un poco más complejo que los ambientes de cartPole o aquellos que manejan una sola dimensión de movimiento. El objetivo de lunar lander as obtener la mayor recompensa posible aterrizando de manera segura el módulo lunar entre las dos banderas amarillas



Hay que tener en cuenta el Action space que está compuesto por 4 posibilidades (no hacer nada,activar motor izquierdo,activar motor principal y activar motor derecho)

El espacio de observación está dado por 8 dimensiones que son :

Coordenadas x y , velocidades lineales en x y , angulo, velocidad angular y dos variables booleanas que representan para cada pierna el contacto con el suelo.

(x,y,vx,vy,theta,omega,right\_leg,left\_leg)

Recompensas

Recompensa por moverse desde la parte superior de la pantalla hasta la plataforma de aterrizaje y detenerse es de aproximadamente 100-140 puntos. Si el módulo de aterrizaje se aleja de la plataforma de aterrizaje, pierde recompensa. Si el módulo de aterrizaje se estrella, recibe una penalización adicional de -100 puntos. Si se detiene, recibe una bonificación adicional de +100 puntos. Cada pata en contacto con el suelo vale +10 puntos. Disparar el motor principal cuesta -0,3 puntos por cada fotograma. Disparar el motor lateral cuesta -0,03 puntos por cada fotograma. Se considera resuelto con 200 puntos

**Enfoque**

Se decidió optar por deepQlearning debido a que es mejor mapear el espacio de observación que se considera continuo hacia un espacio de acción discreto a través de una red neuronal

Se construyó un objeto agente llamado DeepQAgent3 con los siguientes métodos

1. Build\_model: para construir la red neuronal que mapeara la función de valor
2. remember: permitira agregar a su memoria el siguiente estado
3. Choose\_action: escoje la accion basado en epsilon si es por exploración o explotation propagando la red neuronal
4. Learn: permite el entrenamiento de la red neuronal al tiempo que va interactuando con el entorno
5. Adjust\_epsilon: ajusta el valor de epsilon basado en el decremento establecido al inicio del entrenamiento
6. Replay: toma una muestra aleatoria de la memoria preestablecida para entrenarse

Entrenamiento

1. Se establecen 200 episodios con un tamaño de memoria de 10000 observaciones y un batch\_size de 64 para la red neuronal
2. Se llena la memoria del agente de manera prácticamente aleatoria para tener una base de datos de recuerdos con la interacción del ambiente
3. Se procede con el entrenamiento principal

3.1 se escoge la acción ya sea por exploración o explotation

3.2 se pide que el agente recuerde el estado actual, la recompensa por esa acción, la respectiva acción y si ya fracasó o logró aterrizar

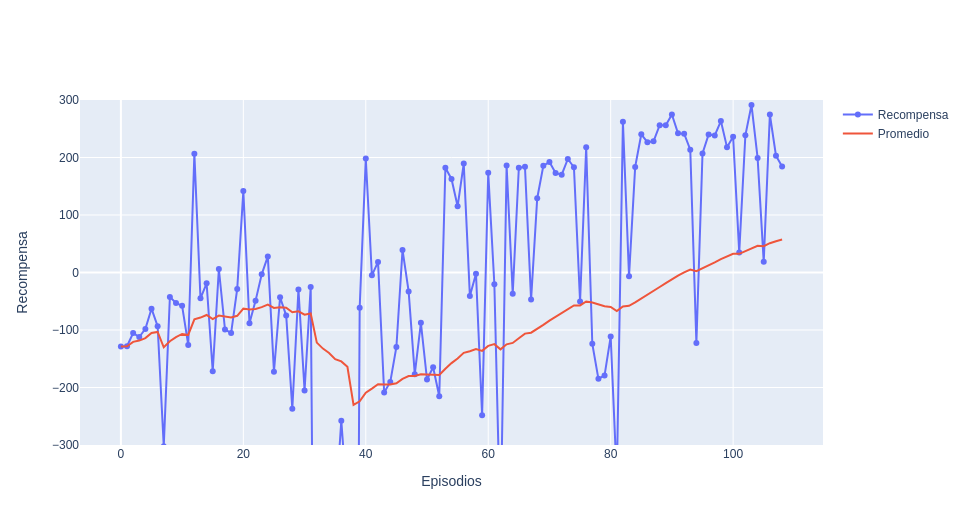
3.3 se obtiene la pérdida del modelo por medio del método replay que invoca el método learn

3.4 se vuelve al paso 3.1 mientras el agente no haya terminado su episodio ya sea por fracaso o éxito

4. si el agente logró aterrizar se guarda su recompensa , se reinicia el estado y se procede con otro episodio

La siguiente gráfica muestra la evolución de las recompensas en un poco más de 40 episodios resolviendo el problema aproximadamente desde el episodio 80

La línea azul es la recompensa por episodio y la línea roja es el promedio móvil de las recompensas (el eje y son las recompensas y el eje x son los episodios)



## 2.3. Manejo de Datos de Alta Dimensionalidad

**Tarea:** Utiliza bases de datos vectoriales como Pinecone o Milvus para almacenar y buscar datos de alta dimensionalidad.

**Requisitos:** Proporciona un ejemplo práctico que demuestre cómo indexar datos, realizar búsquedas eficientes y evaluar el rendimiento de las consultas.

Solución:

Se utilizó Pinecone como base de datos vectorial de acuerdo a la solicitud y se creó el cuadernillo Pinecone.ipyn que se encuentra en la carpeta **Parte2/Pinecone**

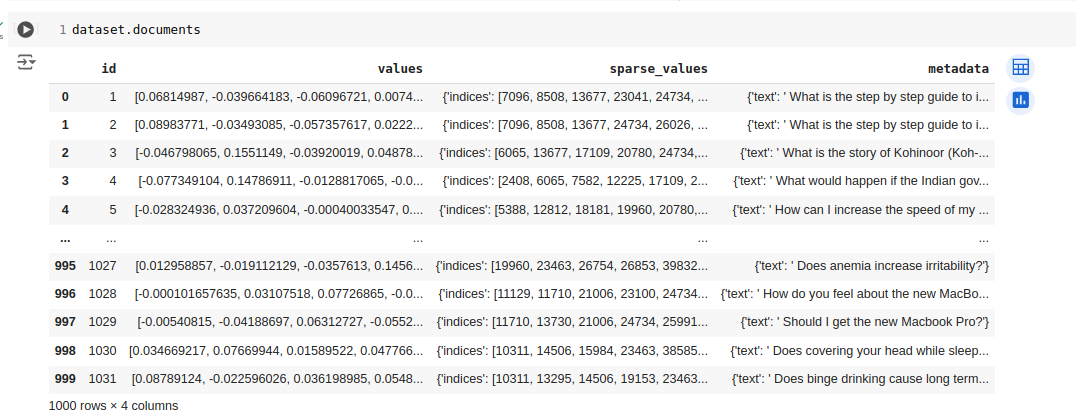
Pasos:

1.Se genera una APIkey desde el panel de contro de Pinecone

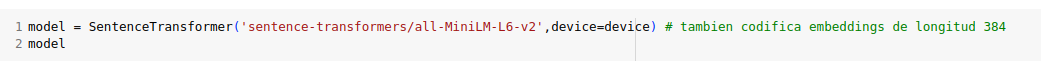
2. Se hacen las respectivas configuraciones



1. Se obtiene un dataset



1. Se utliza un modelo que pueda crear embeddings a partir de texto



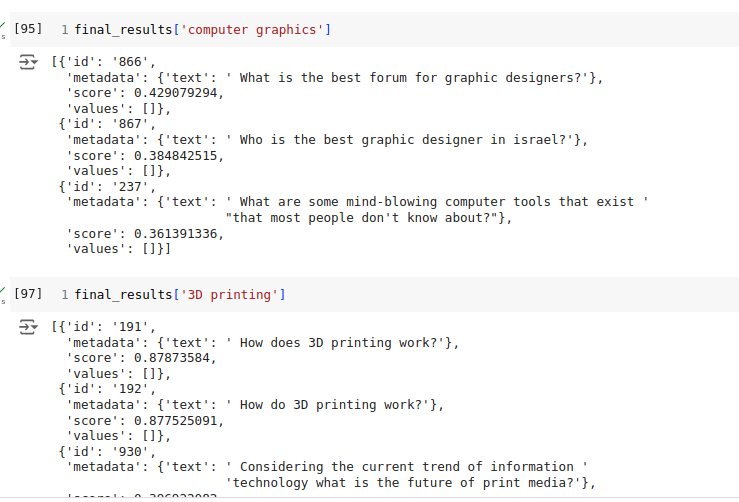
1. Se registra el respectivo index y se inserta la información



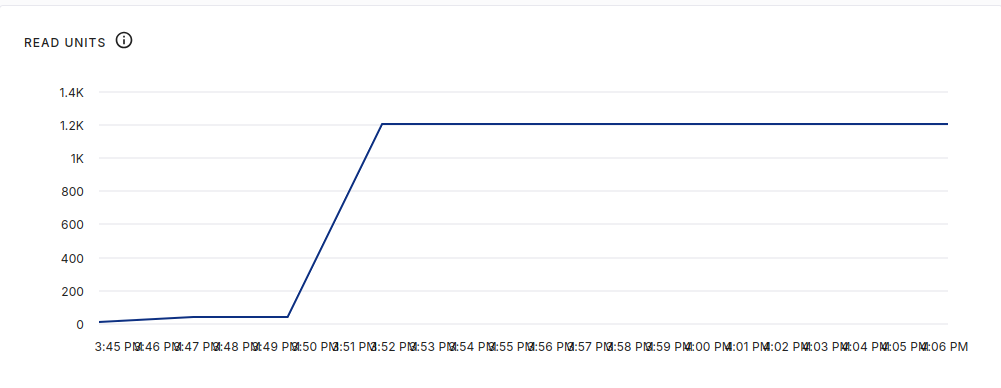
1. Se hacen las respectivas consultas. En este caso se creó una lista aleatoria de varias palabras que serán utilizadas como querys para evaluar el desempeño en el panel de control de pinecone.

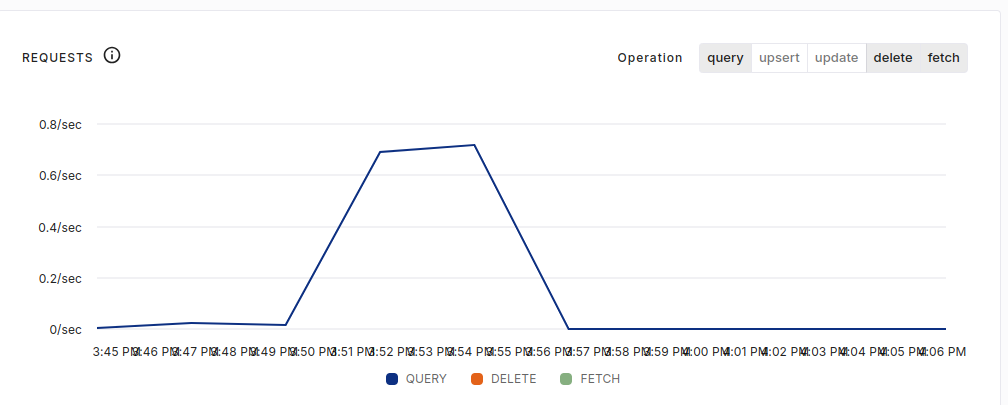


1. Se muestran algunos resultados



Para evaluar el desempeño nos podemos remitir al panel de control de pinecone





De esta manera podemos ver la eficiencia con la que se hacen las consultas, la cantidad de unidades entre otras métricas

# Parte 3: Evaluación de Habilidades de Despliegue

## 3.1. Uso de Docker para Despliegue de Modelos

**Tarea:** Crea un contenedor Docker que incluya un modelo de IA entrenado y una aplicación Streamlit para interactuar con el modelo.

**Requisitos:** Proporciona el Dockerfile, el código de la aplicación, y las instrucciones para ejecutar el contenedor.

**Solución**

El docker file y el código de la aplicación se encuentran en la carpeta **Parte3/dockerapp**

La aplicación usa un modelo sencillo de hugging face de análisis de sentimiento en inglés **cardiffnlp/twitter-roberta-base-sentiment-latest**

Para construir la imagen se tiene el dockerfile, la imagen logo de la app, los requerimientos y el código



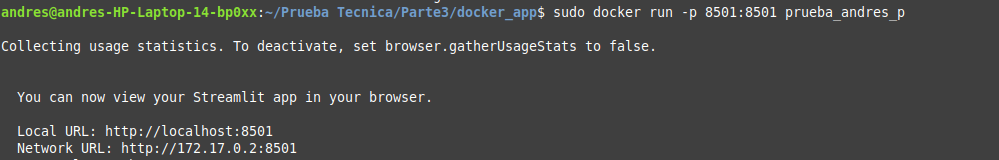
## Se construye la imagen

**andres@andres-HP-Laptop-14-bp0xx:~/Prueba Tecnica/Parte3/docker\_app$** sudo docker build -t prueba\_andres\_p .

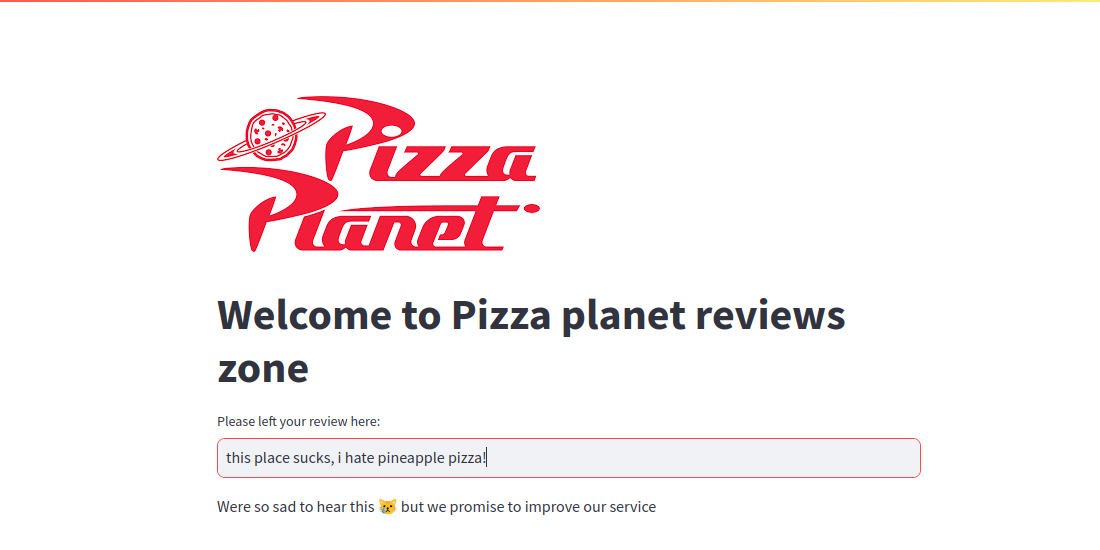
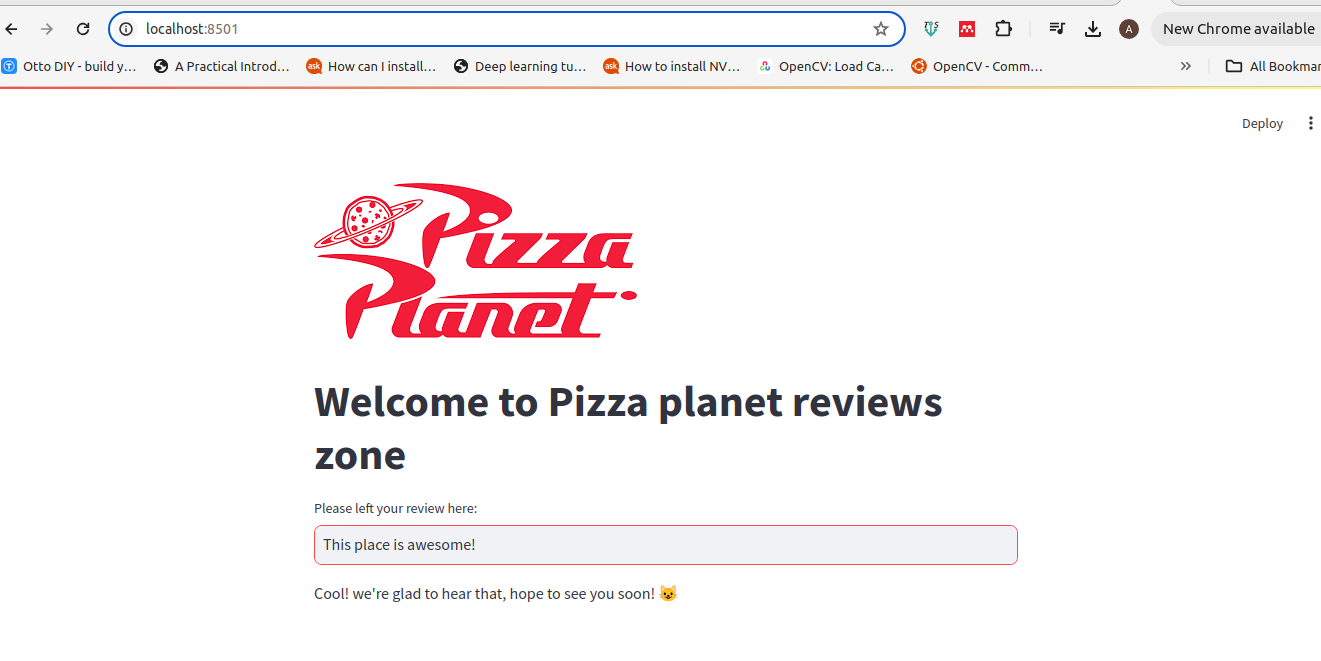


Se activa la imagen

**sudo docker run -p 8501:8501 prueba\_andres\_p**



Se visualiza la app



## 3.2. Implementación de RAGs Personalizados

**Tarea:** Desarrolla un RAG (Retrieval-Augmented Generation) personalizado para mejorar las respuestas generadas por un modelo de lenguaje. Utiliza una base de datos vectorial para la recuperación de documentos relevantes.

**Requisitos:** Explica el enfoque utilizado, implementa el RAG, y proporciona ejemplos de uso con resultados comparativos.

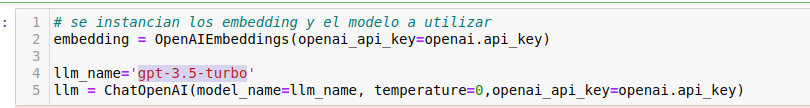
**Solución:**

El código de esta aplicación se encuentra en **/Prueba Tecnica/Parte3/RAG app**

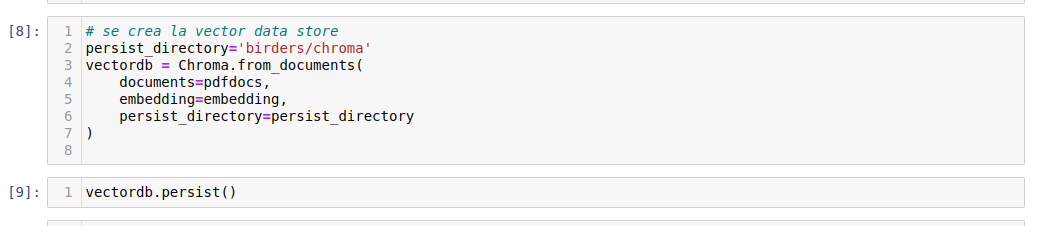
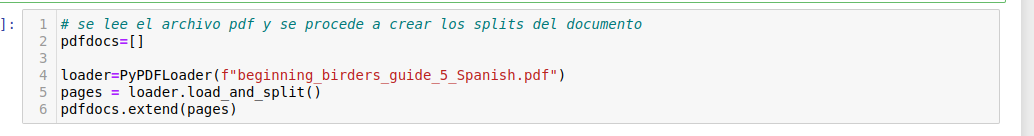
Se desarrolló una interfaz en **Dash plotly** que recibe un vector data store a través de **ChromaDB**

La creación de la base de datos de vectores y métricas se encuentran en el archivo **Vector\_data\_store\_technical\_assesment.ipynb**, como datos de entrada tenemos el documento **beginning\_birders\_guide\_5\_Spanish.pdf**  que es una guía de avistamiento de aves.

El modelo y los embeddings se obtuvierno usando gpt-3.5-turbo a través de la api de openai

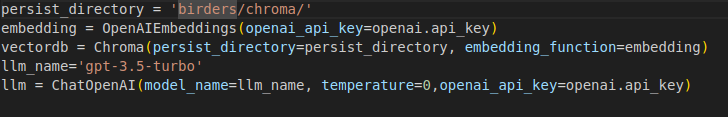


Por medio de langchain se obtuvo la información del respectivo pdf para ser vectorizado en la base de datos



El diseño de la interfaz es un agente chatbot que cuenta con un display de mensajes, un input de texto y un botón para enviar la solicitud. Es importante aclarar que el agente RAG utiliza un objeto de memoria para mantener el contexto de la conversación.

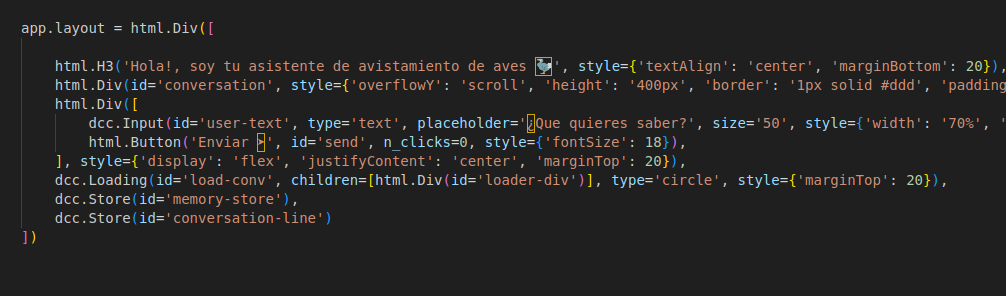
Cargamos la BD vectorizada, con su respectivo modelo



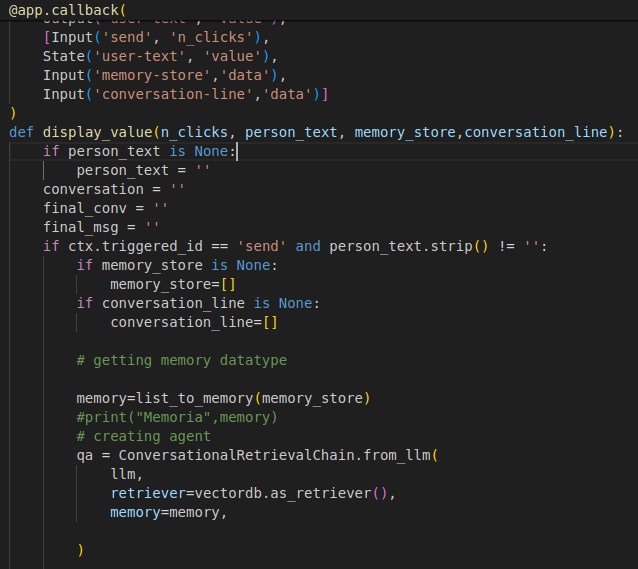
Estas funciones se encargarán de manejar la memoria de la conversación que será almacenada del lado del cliente para no cargar el servidor



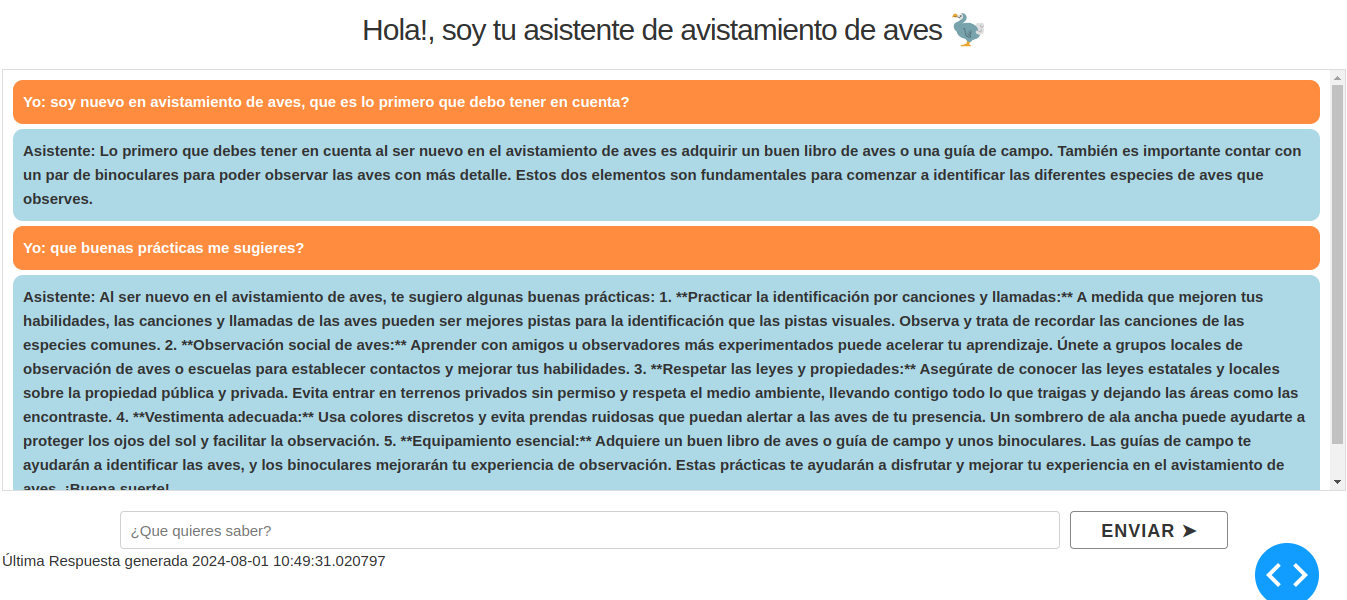
Tenemos el layout de la aplicación

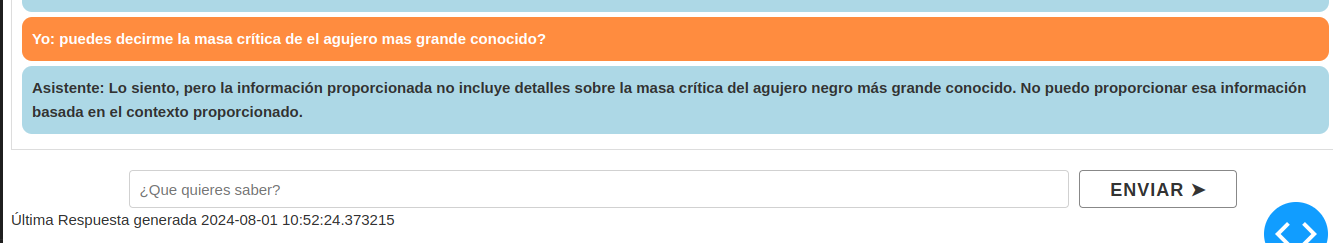


Por último el callback principal

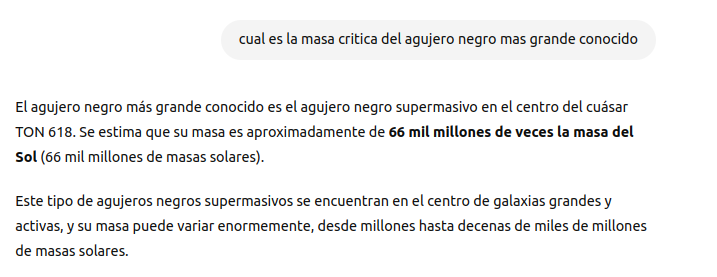


Ejemplo: En este ejemplo se interactúa como una persona principiante en el avistamiento de aves (se puede observar como el bot sigue la línea de conversación)

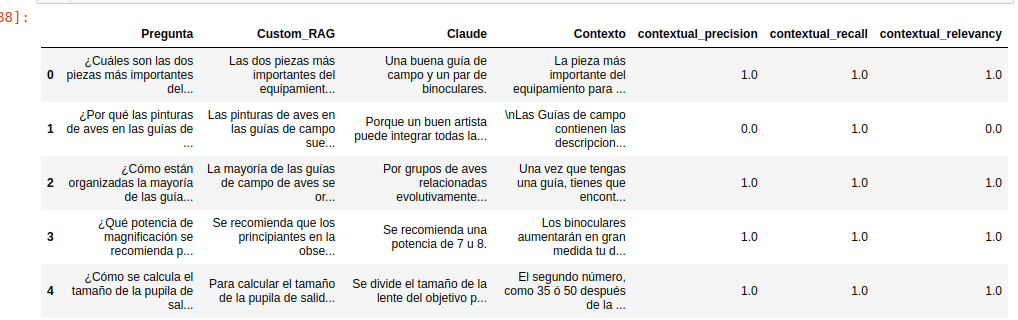
 Ahora que pasa si hago preguntas no relacionadas con la base de datos?



Mientras que el modelo principal



Se evaluó el desempeño de la aplicación comparando con la respuesta de Claude analizando el mismo pdf:



Los puntajes son bastante altos porque estamos comparando dos LLms que tienen alta eficiencia. Lo important es ver que la información para la aplicación rag quedó bien estructurada

Los archivos **ground\_truth** (claude) y **custom\_rag** contienen una serie de preguntas y respuestas dadas por nuestro LLM de referencia y nuestra aplicacióń RAG

# Parte 4: Evaluación de Habilidades de Colaboración y Comunicación

## 4.1. Trabajo en Equipo

**Pregunta:** Describe una experiencia pasada donde colaboraste con un equipo multifuncional para desarrollar una solución de IA. ¿Cuál fue tu papel y cómo contribuiste al éxito del proyecto?

**Respuesta esperada:** Detalles sobre la colaboración, las habilidades técnicas y de comunicación utilizadas, y el impacto del proyecto.

El represamiento de PQRS en mi trabajo siempre ha sido una constante generando tiempos de respuesta lentos y en consecuencia algunas quejas formales se materializan creando inconvenientes para los equipos de atención a los estudiantes.

De acuerdo a esto se realizó un analísis de datos de texto para poder encontrar algunos insights que permitieran ayudar a optimizar algunos procesos dentro de la atención de pqrs de los estudiantes.

De ahi nació la idea de crear un modelo de IA de clasificación de texto que permitiera capturar si una petición de un estudiante manifestaba inconvenientes dentro de su proceso estudiantil, ya sea por temas académicos, financieros, individuales o directamente relacionados con el servicio de la institución. Como primer prototipo se desarrolló un modelo de clasificación Binaria en donde la clase positiva indicaba si el estudiante manifestaba algún problema o inconformidad y la negativa en donde hacia solicitudes que no tenían ningún problema asociado ( solo solicitudes de servicio). Inicialmente experimentamos con una LSTM que entregó resultados bastante aceptables. En ese momento estábamos 3 personas. Un ing de datos, un analista de datos y yo quién actuaba como líder del proyecto y como científico de datos diseñando el pipeline de modelamiento. Actualmente ese modelo está dividido en 2 modelos, el mismo modelo binario detector de problemas pero que ahora funciona con un modelo Roberta. Adicional se incluyó otro modelo Roberta que permite identificar sin importar si la petición es problemática o no el matiz de cada comentario entregando una ponderación de cuál es el factor que más está involucrado en esa petición (financiero, individual, académico , institucional).

En el desarrollo de este proyecto usamos metodología scrum tratando de mantener una comunicación constante a pesar de que los integrantes del proyecto no pertenecemos a una misma área.

Los modelos mencionados anteriormente se desarrollaron en Pytorch y los datos de los comentarios se extrajeron del CRM que serían posteriormente almacenados en un servidor sql a través de la api del mismo CRM.

Una vez terminado el entrenamiento y validación del modelo, este se desplegó a través de una API diseñada en Flask almacenada en la infraestructura interna de la universidad.

La visualización de la clasificación de textos se desarrolló inicialmente sobre un tablero de Power BI que actualmente evolucionó hacia una aplicación analítica desarrollada en Dash plotly organizada por programa académico y segmentado sobre algunos factores sociodemográficos de la institución.

Durante las primeras pruebas se realizó el análisis de los resultados evidenciando que muchos represamientos se daban por algunas solicitudes académicas que requerían intervención del sistema académico por parte parte de admisiones y registro y que en promedio tomaban en ser solucionadas entre 8 y 15 días ocasionando que los estudiantes replicaran la misma solicitud varias veces inflando así la cantidad de solicitudes y demandando más trabajo para las áreas de servicio en su respuesta.

De acuerdo a esto se tomó la decisión de descentralizar algunas de estas solicitudes directamente a los coordinadores de los programas. Finalmente, una vez aplicada esta directriz se evidenció que los tiempos de respuesta en ese tipo de solicitudes bajaron a 3 días y se disminuyó en un 20% la cantidad solicitudes de ese tipo.

## 4.2. Presentación de Resultados

**Tarea:** Prepara una presentación de 10 minutos donde expliques uno de los modelos desarrollados en las tareas anteriores, incluyendo los desafíos encontrados, las soluciones implementadas y los resultados obtenidos.

**Requisitos:** Slides de la presentación y un video o una sesión en vivo presentando el contenido.