Untitled

Carátula

(ver en otras tesinas la información que debe contener la carátula)

Agradecimientos (esto es opcional)

Resumen

Palabras clave: series temporales, predicción, ARIMA, TimeGPT, redes neuronales, ¿?

Tabla de contenidos

Introducción

Contexto de la predicción de series temporales en Argentina. Motivación del uso de modelos avanzados como TimeGPT. Preguntas de investigación y estructura del documento.

Desde antaño, el deseo de saber que traerá el mañana invade los pensamientos de las personas, y tener una predicción del futuro es crucial para la toma de decisiones.

El estudio de las series de tiempo lleva años en desarrollo y permite realizar inferencias en datos temporales de diversas áreas, ya sea finanzas, medicina, medio ambiente u otras tantas más. Con un buen conocimiento de matemáticas, estadística e informática es sencillo hacer un pronóstico aproximado de casi cualquier dato que se mida en el tiempo.

Estos últimos años se vieron caracterizados por el gran aumento en los volúmenes de datos, el '*Big Data*' es presente y futuro, y en un mundo en donde todo se vuelve más complejo y el tiempo es cada vez más valioso, es conveniente tener herramientas que faciliten y acorten los tiempos de trabajo. Si bien los métodos actuales para trabajar series de tiempo son precisos,

los modelos clásicos como ARIMA requieren conocimientos y trabajo manual, mientras que los métodos basados en *machine learning* actuales toman tiempo de entrenamiento y un gran coste computacional. Para resolver estos problemas llega *TimeGPT*, un modelo pre-entrenado exclusivamente para series de tiempo fácil de usar.

Este documento tendrá el trabajo de explicar el funcionamiento de esta nueva tecnología y compararla con otros métodos más asentados en el campo de predicciones de series de tiempo.

Objetivos

El objetivo de esta tesina es, en primer lugar, comparar la precisión, eficiencia y facilidad de pronosticar series de tiempo con *TimeGPT* en contraposición con otros métodos ya más establecidos como los modelos ARIMA o modelos a partir del uso de *machine learning*.

Por otro lado también se busca que el lector obtenga conocimientos acerca de:

- que es una serie de tiempo y para que sirve pronosticarlas
- conceptos básicos del pronóstico de series de tiempo
- como funcionan los modelos de pronóstico utilizados

Para probar y ejemplificar estos puntos es necesario tener datos con los que trabajar, por lo que además de los objetivos planteados anteriormente se propone estimar el consumo mensual de gas natural de la empresa Litoral Gas entre noviembre de 2023 y abril de 2024.

Metodología

Conceptos básicos de series de tiempo

Una serie de tiempo es un conjunto de observaciones ordenadas por el tiempo, generalmente de forma equiespaciada, sobre una variable de interés. El análisis de series de tiempo consiste en resumir y extraer información estadística útil para poder diagnosticar los valores históricos de la variable de interés y pronosticar los futuros.

El gran volúmen de producción de este tipo de datos dado al auge del *Big Data* hace que saber entederlos y analizarlos correctamente sea esencial en el mundo de la estadística.

Las series de tiempo se pueden presentar con distintos patrones, por lo que es de utilidad 'dividir' la serie en distintas componentes, en las que cada una representa una categoría del patrón de la serie:

- La componente estacional reconoce los patrones que se repiten en cada intervalo de tiempo, por ejemplo la temperatura, que baja en invierno y en verano sube repitiendo este patrón cada año.
- La componente tendencia-ciclo (de forma abreviada simplemente tendencia) reconoce como se ve afectada la media a medida que avanza el tiempo, por ejemplo la pobación mundial, que crece cada año.

Se dice que una serie es débilmente estacionaria si la media y la variancia se mantienen constantes en el tiempo y la correlación entre distintas observaciones solo depende de la distancia en el tiempo entre estas. Por comodidad, cuando se mencione estacionariedad se estará haciendo referencia al cumplimiento de estas propiedades.

Se denomina función de autocorrelación a la función de los rezagos, entendiendo por rezago a la distancia ordinal entre dos observaciones, que grafica la autocorrelación entre pares de observaciones. Es decir que para cada valor k se tiene la correlación entre todos los pares de observaciones a k observaciones de distancia. En su lugar, la función de autocorrelación parcial calcula la correlación condicional de los pares de observaciones, removiendo la dependencia lineal de estas observaciones con las que se encuentran entre estas.

Introducción a los modelos transformadores

En los últimos años '*ChatGPT*' ganó una popularidad excesiva a nivel mundial, y rápidamente muchos competidores salieron a la luz para agarrar una parte del gran mercado de la inteligencia artificial. El chatbot de OpenAI hizo que muchas las empresas de tecnología quieran crear su propia versión y usar la tecnología implementada para distintos usos. Las siglas *GPT* provienen de *Generative Pre-trained Transformer*, un tipo de inteligencia artificial de la familia de las redes neuronales que usa arquitectura *transformer*. Entender que significa esto no es tarea sencilla si no se tiene ya una base de conocimientos en el tema, es por eso que a continuación se explican brevemente ciertos conceptos para luego dar pie a los modelos transformadores.

- Inteligencia Artificial (IA): Es la ciencia que busca construir computadoras, máquinas y programas que puedan razonar, aprender y actuar de la misma manera que lo haría un ser humano.
- *Machine Learning* (ML): Es una rama de la inteligencia artificial enfocada a permitir que las computadoras y máquinas imiten la forma en que los humanos aprenden, para realizar tareas de forma autónoma y mejorar la eficiencia y eficacia a través de la experiencia y la exposición a mas información.
- Red Neuronal (*neuronal network*): Las redes neuronales son un tipo de programa o modelo de *machine learning* que toma decisiones de la misma forma que las personas, usando procesos que simulan la forma biologica en la que trabajan la neuronas para

indentificar fenómenos, evaluar opciones y llegar a conclusiones. **EXTENDER CON BREVE EXPLICACION SOBRE CAPAS Y TIPOS COMO CNN Y RNN**

- Aprendizaje profundo (*Deep learning*): Es una rama del *machine learning* que tiene como base un conjuntos de algoritmos (entre ellos las redes neuronales) que intentan modelar niveles altos de abstracción en los datos usando múltiples capas de procesamiento, con complejas estructuras o compuestas de varias transformaciones no lineales.

Los modelos transformadores (*transformer models*) son un tipo de modelos con arquitectura de aprendizaje profundo. Su auge se ve explicado por su alta eficiencia para entrenar y realizar inferencias en comparación con otros métodos de aprendizaje profundo tales como las redes neuronales recurrentes (RNN) o redes neuronales convolucionales (CNN). Esta eficiencia se debe al uso de mecanismos de atención, presentados en la publicación '*Attention is all you need*' de Google, este mecanismo se detallará más adelante para poder ser seguido con ejemplos.

Modelos utlilizados para pronosticar las series temporales

ARIMA

Los modelos $ARIMA$ (*AutoRegresive Integrated Moving Average*) son unos de los modelos de pronostico tradicionales mejor establecidos. Son una generalización de los modelos autoregresivos (AR), que suponen que las observaciones futuras son función de las observaciones pasadas, y los modelos promedio móvil (MA), que pronostican las observaciones como funciones de los errores de observaciones pasadas. Además generaliza en el sentido de los modelos diferenciados (I), en los que se resta a cada observacion los d-ésimo valores anteriores para estacionarizar en media.

Formalmente un modelo $ARIMA(p, d, q)$ se define como:

$$\psi_p(B)(1-B)^d Z_t = \theta_0 + \theta_q(B)\alpha_t$$

Donde Z_t es la observación t-ésima, $\psi_p(B)$ y $\theta_q(B)$ son funciones de los rezagos (B), correspondientes a la parte autoregresiva y promedio móvil respectivamente, d es el grado de diferenciación y α_t es el error de la t-ésima observación.

Se debe tener en cuenta estos aspectos importantes:

- Se dice que una serie es invertible si se puede escribir cada observación como una función de las observaciones pasadas más un error aleatorio. Por definición, todo modelo AR es invertible.
- Por definición, todo modelo MA es estacionario.

- $\psi_p(B) = 1 - \psi_1 B - \psi_2 B_2 - ... - \psi_p B^p$ es el polinomio característico de la componente AR y $\theta_q(B) = 1 - \theta_1 B - \theta_2 B^2 - ... - \theta_p B^q$ de la componente MA. Si las raíces de los polinomios característicos caen fuera del círculo unitario, entonces un proceso AR se puede esctribir de forma MA y es estacionario, y a su vez un proceso MA se puede escribir de forma AR y es invertible.
- Un proceso $ARIMA$ es estacionario e invertible si su componente AR y MA lo son respectivamente.

Sin embargo este tipo de modelos no tienen en cuenta la posible estacionalidad que puede tener una serie, es por esto que se introducen los modelos $SARIMA(p, d, q)(P, D, Q)_s$ que agregan componentes AR, MA y diferenciaciones a la parte estacional de la serie con período s.

Un buen modelo $SARIMA$ debe cumplir las siguientes propiedades:

- Sus residuos se comportan como ruido blanco, es decir, estan incorrelacionados y se distribuyen normal, con media y variancia constante.
- Es admisible, es decir que es invertible y estacionario.
- Es parsimonioso, en el sentido de que sus parámetros son significativos.
- Es estable en los parámetros, que se cumple cuando las correlaciones entre los parámetros no son altas.

TimeGPt

TimeGPT es el primer modelo fundacional (modelos entrenados con conjuntos de datos masivos)[1] pre-entrenado para el pronóstico de series de tiempo que puede producir predicciones en diversas áreas y aplicaciones con gran precisión y sin entrenamiento adicional. Los modelos pre-entrenados constituyen una gran innovación haciendo que el pronóstico de series de tiempo sea más accesible, preciso, tenga menor complejidad computacional y consuma menos tiempo.

TimeGPT es un modelo de tipo transformer con mecanismos de autoatención (del inglés *self-attention mechanisms*) que **no** es de código abierto. La autoatención captura dependencias y relaciónes en la secuencias de valores que se alimentan al modelo, logrando poner en contexto a cada observación.

La arquitectura del modelo consiste en una estructura con codificador y decodificador de múltiples capas, cada una con conexiones residuales y normalización de capas. Por último, contiene una capa lineal que mapea la salida del decodificador a la dimensión del pronóstico.

[1] Según los creadores de TimeGPT, fue entrenado con una colección de 100 mil millones de observaciones provenientes de datos públicos.

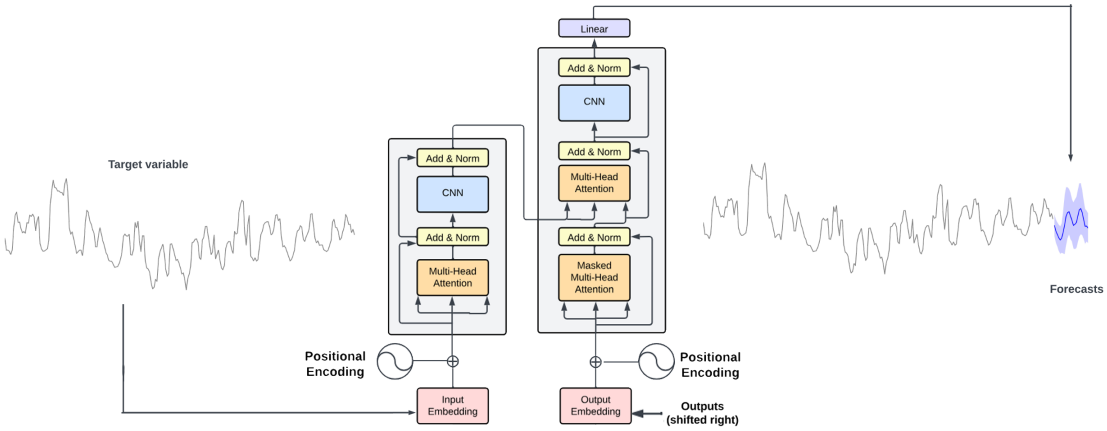


Figure 1: Diagrama de la estructura del modelo

Paso a paso del funcionamiento del modelo

Codificador

1. *Input embedding*: Cada *token* (observaciones en el caso de TimeGPT) es transformado en un vector (vector de entrada) con muchas dimensiones ($\vec{E}$). Las dimensiones corresponden a diferentes caracteristicas que el modelo definió en el pre-entrenado con una numerosa cantidad de parámetros.

2. Codificación posicional: Típicamente se introduce como un set de valores adicionales o vectores que son añadidos a los vectores de entrada antes de alimentarlos al modelo ($\vec{E} \Leftarrow \vec{E} + \vec{P}$). Estas codificaciones posicionales tienen patrones específicos que agregan la información posicional del token.

3. *Multi-Head Attention* (Atención multi-cabezal): La autoatención opera en múltiples 'cabezales de atención' para capturar los diferentes tipos de relaciones entre tokens. Una cabeza de atención verifica el contexto en el que el token está y manipula los valores del vector que lo representa para añadir esta información contextual.

La verificación del contexto funciona gracias a una matriz la cúal se llamará *Query* (W_Q) que examina ciertas características, y es multiplicada por el vector de entrada ($\vec{E}$) resultando en un vector de consultas ($\vec{Q}$) para cada token. Una matriz de claves (*Key matrix*) (W_K) que comprueba las relaciones con las características en la matriz de consultas y tiene las mismas dimensiones que esta es también multiplicada por $\vec{E}$ generando así el vector de claves ($\vec{K}$). Luego se forma una nueva matriz a partir de los productos cruzados entre los vectores $\vec{K}$ y

$\vec{Q}$ de cada token, se divide por la raíz de la dimensión de los vectores[2] ($\sqrt{d_k}$) y se normaliza con *softmax*[3] por columna[4] ($\vec{S}$), valores más altos indican que un token (de las columnas) esta siendo influenciado por el comportamiento de otro token (de las filas).

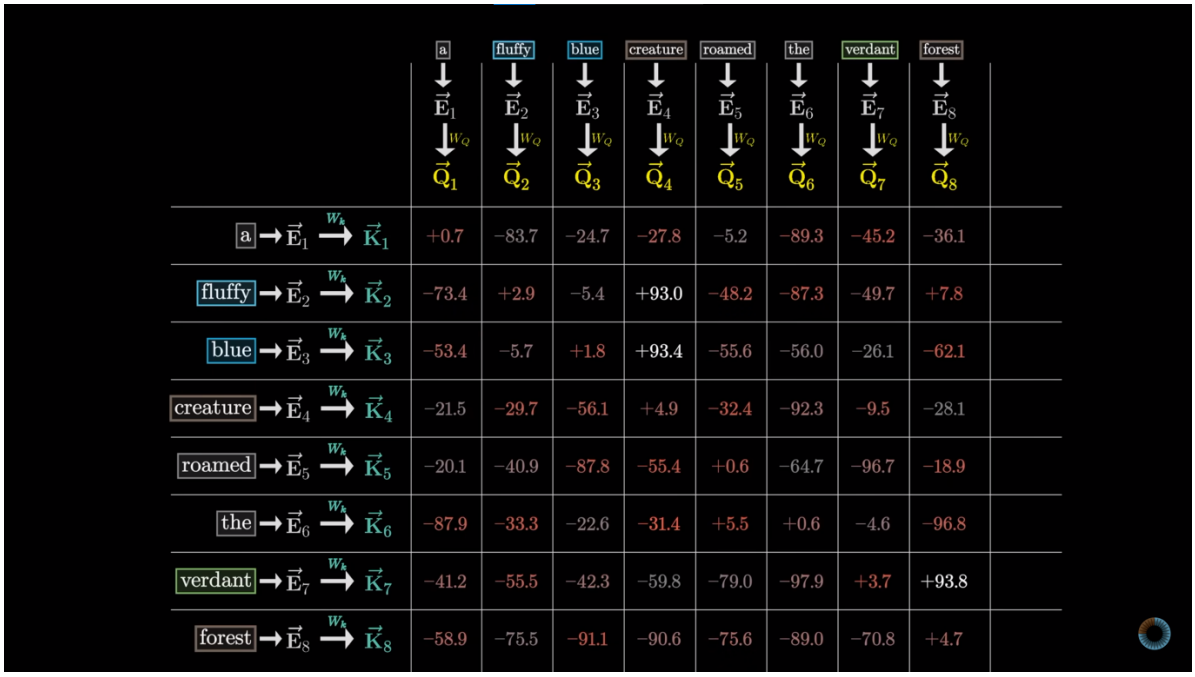


Figure 2: (before applying softmax) for example: Fluffy and Blue contribute to creature

Ahora que se sabe que tokens son relevantes para otros tokens, es necesario saber como son afectados. Para esto existe otra matriz de valores (W_V) que es multiplicada por cada vector de entrada resultando en los vectores de valor ($\vec{V}$) que son multiplicados a cada columna. La suma por filas devuelven el vector ($\Delta\vec{E}$) que debe ser sumado al vector de entrada original de cada token.

[2]Es útil para mantener estabilidad numérica, la cual describe cómo los errores en los datos de entrada se propagan a través del algoritmo. En un método estable, los errores debidos a las aproximaciones se atenúan a medida que la computación procede.

[3]$softmax(x) = \frac{e^{x_i/t}}{\sum_j e^{x_j/t}}$

[4]Aplicar *softmax* hace que cada columna se comporte como una distribución de probabilidad.

$$\Delta\vec{E}_j = \sum_i S_j \cdot \vec{V}_i$$

Con múltiples 'cabezas', cada una con sus propias matrices W_K, W_Q y W_V, se generan varios $\Delta\vec{E}$ que se suman y se añaden al vector de entrada original.

$$\vec{E} \Leftarrow \vec{E} + \sum_h \Delta\vec{E}_h$$

En resumen, y haciendo referencia la publicación Attention is all you need:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

Es importante notar que todas las matrices Q, K y V son pre-entrenadas.

4. Sumar y normalizar (*Add and norm*): En lugar de simplemente pasar los datos por las capas, las conexiones residuales se añaden sobre el vector de entrada en la salida de cada capa. Esto es lo que se hizo cuando se sumaron los cambios al vector de entrada, en lugar de directamente modificar el vector.

Dado que las redes neuronales profundas sufren de inestabilidad en los pesos al actualizarlos, una normalización al vector estabiliza el entrenamiento y mejora la convergencia.

5. Red neuronal convolucional (CNN): A diferencia de otros modelos transformadores tradicionales, TimeGPT incorpora *CNN*s para descubrir dependencias locales y patrones de corto plazo, tratando de minimizar una función de costo (MAPE, MAE, RMSE, etc.).

Decodificador

1. *Output embedding* (desplazado hacia la derecha): La entrada del decodificador son los tokens desplazados hacia la derecha.
2. Codificación posicional
3. *Masked multi-head attention* (Atención multicabezal enmascarada): Ahora que las predicciones deben hacerce únicamente con los valores previos a cada token, en el proceso de 'atención' y antes de aplicar la transformación softmax se debe reemplazar todos los valores debajo de la diagonal principal de la matriz QK por $-\infty$ para prevenir que los tokens sean influenciados por tokens anteriores.
4. *Multi-head attention*: En este caso, se usan las matrices de claves y valores que da como salida el codificador y la matriz de consultas es la salida de la capa de atención multicabezal enmascarada.
5. Conexión lineal: Es una capa completamente conectada que traduce las representaciones de atributos aprendidas en predicciones relevantes.

Métricas de evaluación

Es muy posible que más de un modelo ajuste bien, por eso es necesario tener alguna forma de compararlos para elegir el mejor con un criterio formal. Para esto se usan medidas del error sobre los valores pronosticados.

Sea $e_l = Z_{n+l} - \hat{Z}_n(l)$ el error de la l-ésima predicción, donde $\hat{Z}_n(l)$ es el pronóstico l pasos hacia adelante, algunas medidas del error para pronósticos L pasos hacia adelante son:

- Error Cuadrático Medio (*Mean Square Error (MSE)*):

$$MSE = \frac{1}{L}\sum_{l=1}^{L} e_l^2$$

- Error absoluto medio (*Mean Absolute Error (MAE)*):

$$MAE = \frac{1}{L}\sum_{l=1}^{L} |e_l|$$

- Porcentaje del error absoluto medio (*Mean Absolute Percentage Error (MAPE)*)

$$MAPE = (\frac{1}{L}\sum_{l=1}^{L}|\frac{e_l}{Z_{n+l}}|) \cdot 100\%$$

Aquel modelo que tenga el menor valor en estas medidas será considerado el mejor.

El problema de estos errores es que solo tienen en cuenta la estimación puntual, y por lo general es buena idea trabajar con pronosticos probabilisticos para cuantificar la incertidumbre de los valores futuros de la variable. Gneiting & Raftery (2007, JASA) propusieron en Strictly Proper Scoring Rules, Prediction, and Estimation una nueva medida del error que tiene en cuenta los intervalos probabilísticos de la estimación, llamandola *Interval Score*:

$$S = \frac{1}{L}\sum_{l=1}^{L}(W_l + O_l + U_l)$$

Donde:

$$W_l = IS_l - II_l$$

$$O_l = \begin{cases} \frac{2}{\alpha}(Z_n(l) - Z_{n+l}) & \text{si } Z_n(l) > Z_{n+l} \\ 0 & \text{en otro caso} \end{cases} \qquad U_l = \begin{cases} \frac{2}{\alpha}(Z_{n+l} - Z_n(l)) & \text{si } Z_n(l) < Z_{n+l} \\ 0 & \text{en otro caso} \end{cases}$$

Siendo IS_l e II_l los extremos superior e inferior del intervalo del l-ésimo pronóstico respectivamente. Es fácil darse cuenta que W es una penalización por el ancho del intervalo, y que O y U son penalizaciones por sobre y subestimación respectivamente.

Aplicación

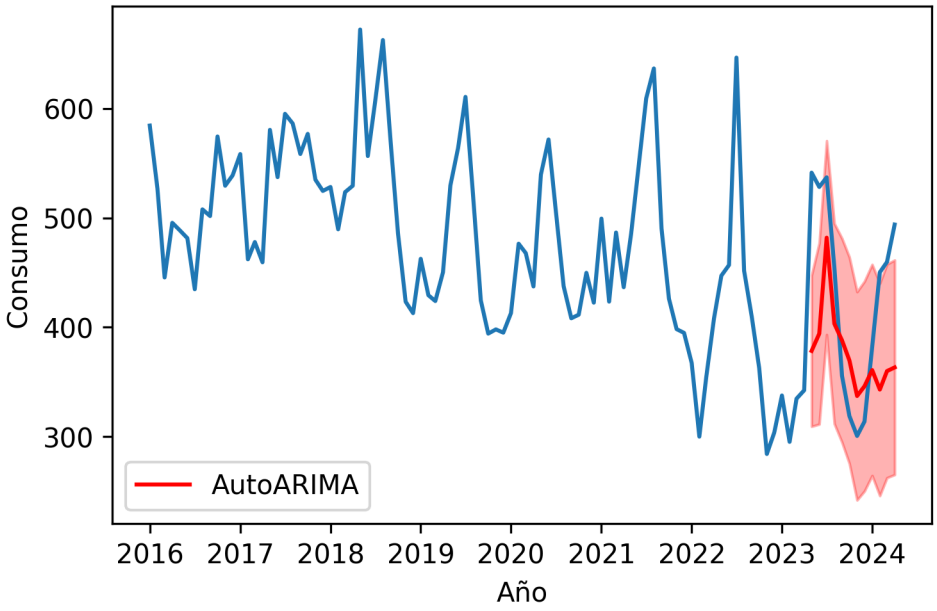
- Selección de la serie temporal: Descripción de la serie mensual utilizada (ejemplo: inflación, exportaciones, producción industrial, etc.).
- Tratamiento de datos: Limpieza, detección de valores atípicos y transformación de variables.
- Implementación de modelos: Configuración y parámetros utilizados en ARIMA, ML y TimeGPT.
- Criterios de evaluación: Explicación de métricas utilizadas (RMSE, MAE, MASE, etc.).

Resultados y Comparación de Modelos

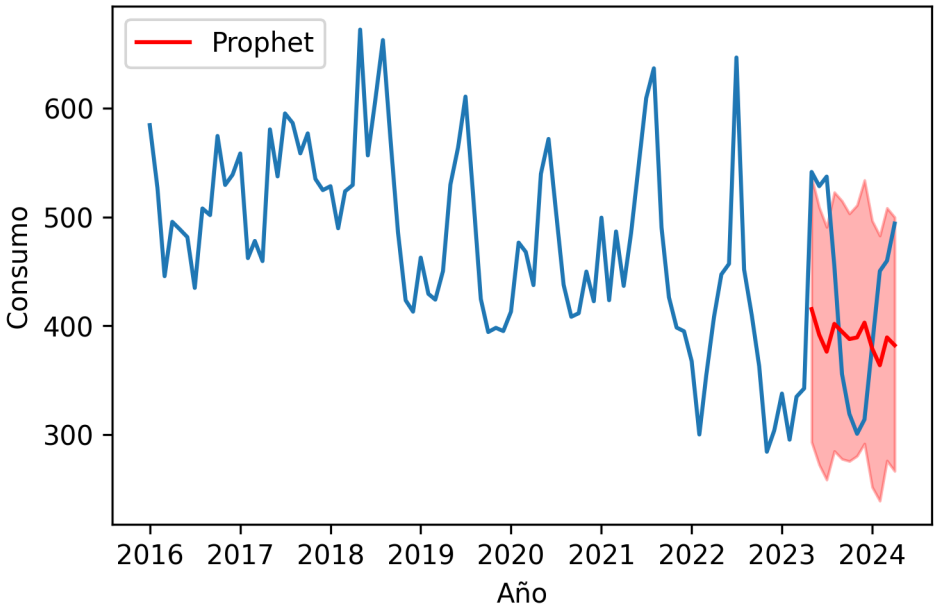
- Descripción de los resultados obtenidos con cada modelo.
- Comparación de desempeño en distintos horizontes de predicción.

Métodos Tradicionales

ARIMA

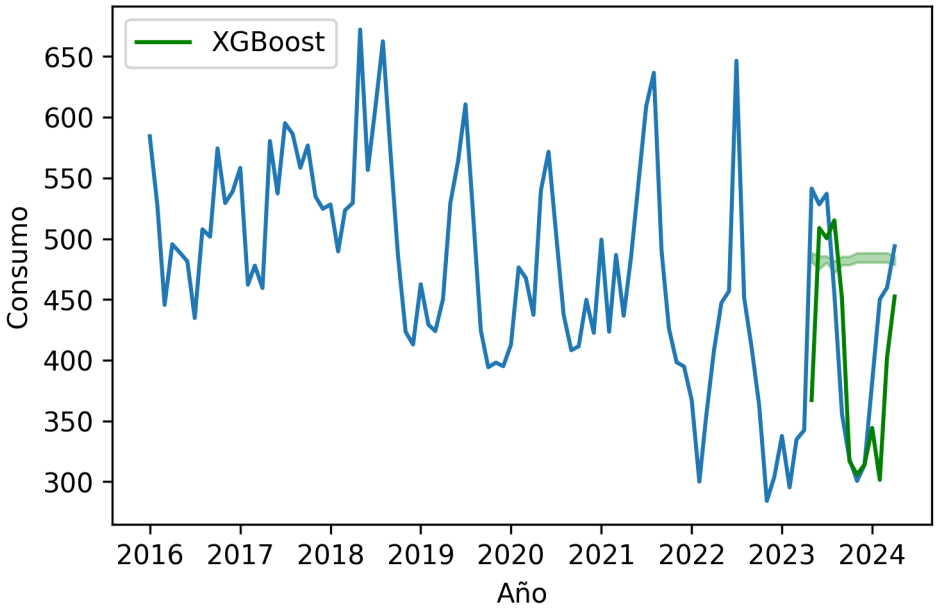


Prophet

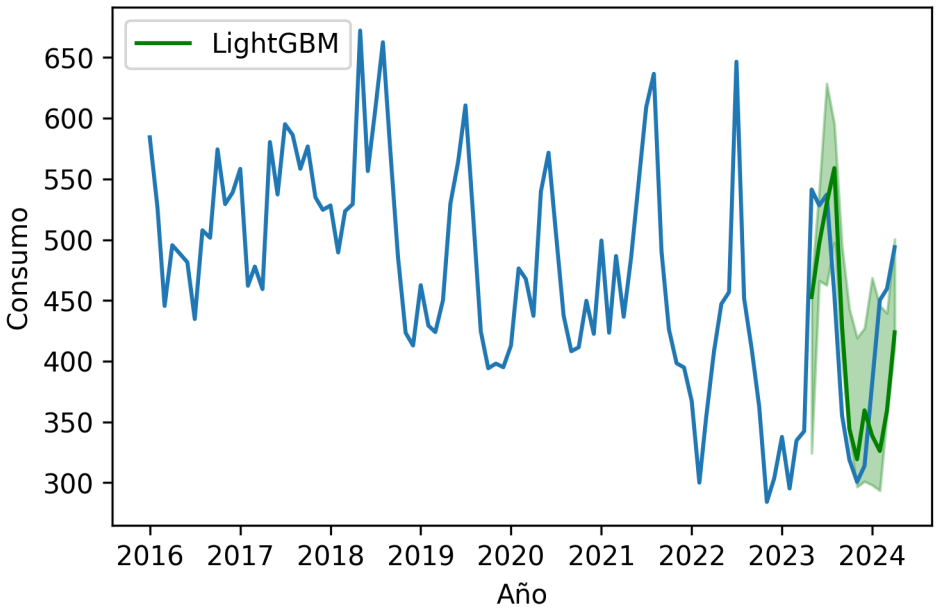


Métodos de Machine Learning

XGBoost

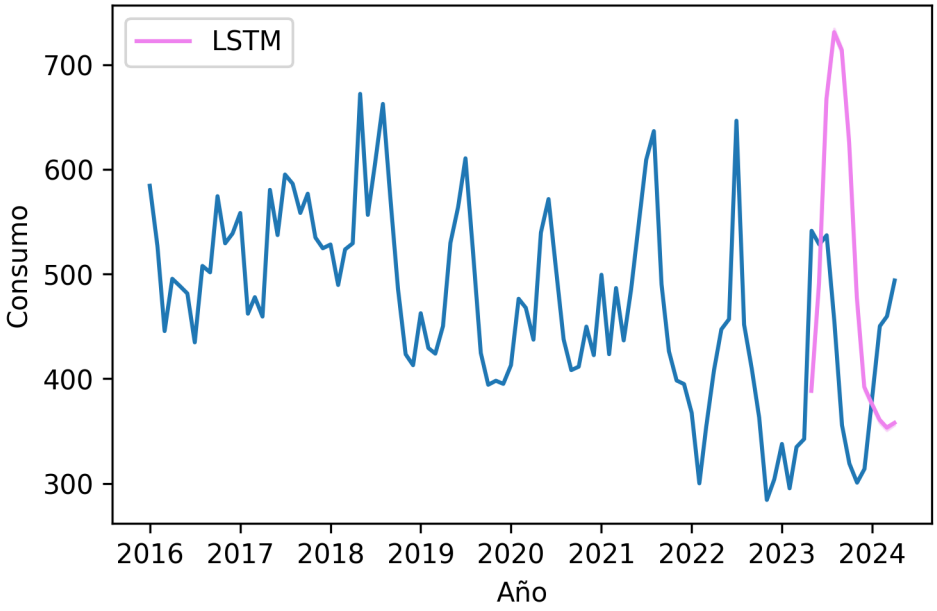


LightGBM

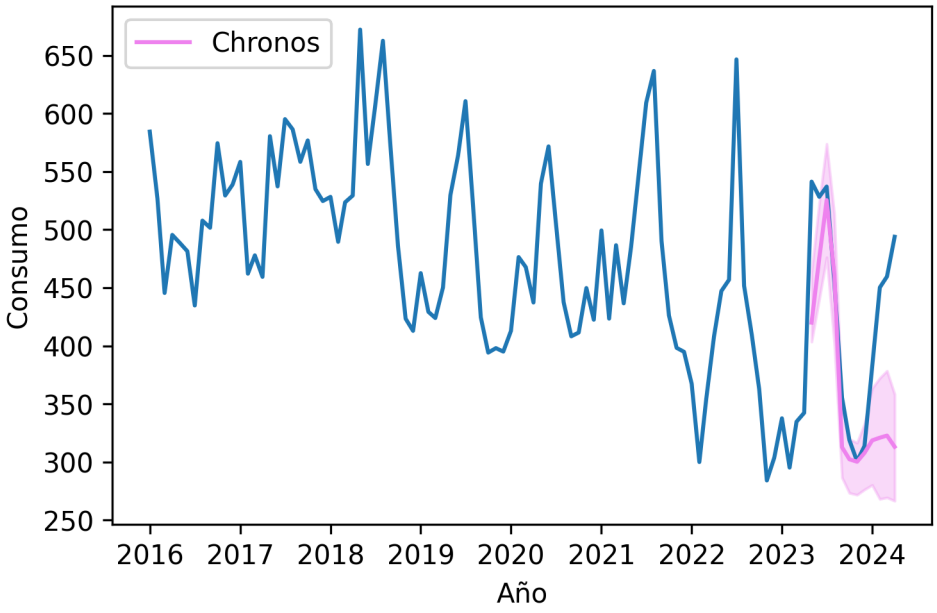


Deep Learning

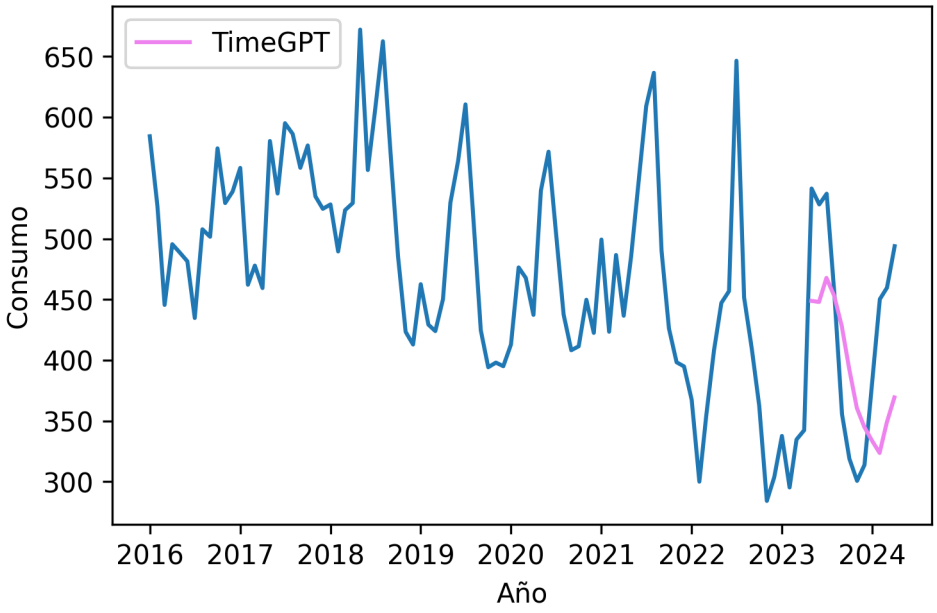
LSTM



Chronos



TimeGPT



Comparaciones

	Modelo	MAPE	Interval Score	Tiempo
0	AutoARIMA	0.168367	341.615346	13.664841
1	Prophet	0.199253	295.883113	1.712066
2	XGBoost	0.123905	800.897797	0.138012
3	LightGBM	0.142309	290.805218	1.546838
4	LSTM	0.394279	1548.627002	11.119949
5	TimeGPT	0.174341	740.443642	2.151767
6	Chronos	0.139081	415.731862	2.151767

Conclusiones

- Resume las principales conclusiones del estudio.
- Destaca las contribuciones del trabajo y su relevancia para el campo de la estadística y el análisis de datos.
- Fortalezas y limitaciones de TimeGPT en comparación con otros modelos.
- Proporciona recomendaciones finales y reflexiones sobre posibles direcciones futuras de investigación.

Referencias Bibliográficas

Ir completando a medida que vas consultando artículos o libros

Anexo

Incluye cualquier material adicional, como código fuente, datos adicionales, detalles sobre la implementación en R o Python, entre otros.

IDEAS

Distintos modelos a usar para el apartado machine learning

- Guardar las versiones de librerias, software y hardwate en la que se corre el codigo
- Hacer equivalencia del documento en GitBook

- Enfocar la tesina a comparar modelos tradicionales vs machine learning vs deep learning
- Datos: Graciela N. Klekailo Facultad de Ciencias Agrarias, Direccion general de estadistica municipalidad
- Investigar mas sobre medidas de error
- Toda la modelización pasarla a .py y no .qmd
- Implementar interval_score como metrica para chronos y timegpt?
- Modelos que voy a usar: Tradicionales: Arima y autoarima, Prophet Machine Learning: XGB, LGBM Deep Learning: LSTM Foundational models: TimeGPT, Chronos
- leer fpp3 que tiene unas cosas que pueden servir