UNIVERSIDAD NACIONAL DEL ALTIPLANO

FACULTAD DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

ESCUELA PROFESIONAL DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

IMPLEMENTACIÓN DE UN SISTEMA INFORMÁTICO INTELIGENTE PARA EL PRONÓSTICO DE VENTAS DE LA CEVICHERÍA RICO NORTE DE LA CIUDAD DE PUNO 2024

TESIS

PRESENTADA POR:

Bach. RODRIGO ALEXANDER BECERRA LUCANO

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO ESTADÍSTICO E INFORMÁTICO

PUNO - PERÚ

2024

Reporte de similitud

NOMBRE DEL TRABAJO

IMPLEMENTACIÓN DE UN SISTEMA INFORMÁTICO INTELIGENTE PARA EL PRONÓSTICO DE VENTAS DE LA CEVICHERÍA R

AUTOR

RODRIGO ALEXANDER BECERRA LUCANO

RECUENTO DE PALABRAS

22280 Words

RECUENTO DE CARACTERES

144089 Characters

RECUENTO DE PÁGINAS

157 Pages

TAMAÑO DEL ARCHIVO

5.5MB

FECHA DE ENTREGA

Oct 3, 2024 7:40 PM GMT-5

FECHA DEL INFORME

Oct 3, 2024 7:44 PM GMT-5

14% de similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para cada base de datos.

- 9% Base de datos de Internet
- 5% Base de datos de publicaciones
- Base de datos de Crossref
- Base de datos de contenido publicado de Crossref
- 13% Base de datos de trabajos entregados

Excluir del Reporte de Similitud

- Material bibliográfico
- Material citado
- Material citado
- Coincidencia baja (menos de 10 palabras)

Universidad Nacional del Altiplano de Puno
Firmado digitalmente por JUAREZ VARGAS Juan Carlos FAU 20145498170 soft
Motivo: Soy el autor del documento
Fecha: 03.10.2024 19:46:49 -05:00

Universidad Nacional del Altiplano de Puno
Firmado digitalmente por JUAREZ VARGAS Juan Carlos FAU 20145498170 soft
Motivo: Soy el autor del documento
Fecha: 03.10.2024 19:47:00 -05:00

Resumen

DEDICATORIA

A mi madre Valentina y a mi padre Paco, quienes fueron mi aliento y apoyo en este proceso y en todo el transcurso de mi vida, también por su amor, consejos y siempre haber creído en mí. Es gracias a ellos que tuve la oportunidad de ser un profesional.

A mis hermanas Magaly y Yésica, por confiar en mí y por su apoyo en momentos importantes de mi vida.

A mi enamorada Thania, por su amor, compresión y apoyo, sobre todo en los momentos difíciles.

Rodrigo Alexander B. L

AGRADECIMIENTOS

A mi alma mater, la Universidad Nacional del Altiplano y a la facultad de Ingeniería Estadística e Informática por el conocimiento y experiencia necesarias que me permitieron llegar hasta aquí y darme la posibilidad de ser un gran profesional.

A mi asesor de tesis D. Sc. Juan Carlos Juárez Vargas, por su guía en mi etapa universitaria y en este trabajo de investigación.

Rodrigo Alexander B. L

ÍNDICE GENERAL

Pág.

ÁREA: Sistemas, Computación e Informática

TEMA: Ingeniería de Software e Inteligencia Artificial

FECHA DE SUSTENTACIÓN: 9 de octubre del 2024

ÍNDICE DE TABLAS

Pág.

ÍNDICE DE FIGURAS

Pág.

ÍNDICE DE ANEXOS

Pág.

ACRÓNIMOS

CRISP-DM:	Proceso Estándar de Minería de Datos para Industrias Cruzadas (Cross-Industry Standard Process for Data Mining)
XGBOOST:	Impulso de Gradiente Extremo (eXtreme Gradient Boosting)
SARIMA:	Modelo Autorregresivo Integrado de Medias Móviles Estacional (Seasonal Autoregressive Integrated Moving Average)
LSTM:	Memoria a Largo – Corto Plazo (Long Short-Term Memory)
MLP:	Perceptrón Multicapa (Multilayer Perceptron)
RNN:	Red Neuronal Recurrente (Recurrent Neural Network)
MAE:	Error Absoluto Medio (Mean Absolute Error)
MSE:	Error Cuadrático Medio (Mean Squared Error)
R^2:	Coeficiente de Determinación
MAPE:	Error Porcentual Absoluto Medio (Mean Absolute Percentage Error)
HTML:	Lenguaje de Marcas de Hipertexto (HyperText Markup Language)
CSS:	Hojas de Estilo en Cascada (Cascading Style Sheets)
SQL:	Lenguaje de Consulta Estructurada (Structured Query Language)
CSV:	Valores Separados por Comas (Comma Separated Values)

RESUMEN

La presente tesis tuvo como objetivo implementar un sistema informático inteligente que pronostique las ventas diarias de la cevichería Rico Norte de la ciudad de Puno, Perú, como solución a problemas como la falta y desperdicio de insumos, la tardía atención en días con mucha clientela, y la necesidad de realizar cambios en el negocio que no perjudiquen su economía. La investigación se llevó a cabo en los años 2023 y 2024. Se usó la metodología de desarrollo CRISP-DM que consta de 6 fases. En la fase de comprensión del negocio se abarcaron las necesidades y requerimientos. En la comprensión de datos se recopilaron los datos proporcionados por el dueño. En la preparación de datos se realizó el preprocesamiento de datos. En el modelado se implementaron los modelos de Machine Learning SARIMA (Modelo Autorregresivo Integrado de Media Móvil Estacional), XGBoost (Extreme Gradient Boosting), LSTM (Long Short-Term Memory), y MLP (Modelo de Perceptrón Multicapa). En la evaluación se usaron las métricas de rendimiento MAE (Error Absoluto Medio), MSE (Error Cuadrático Medio), R^2 (Coeficiente de Determinación), y MAPE (Error Porcentual Absoluto Medio). En el despliegue se implementó el sistema con las tecnologías HTML, CSS, JavaScript, Python y SQL, integrando el modelo XGBoost debido a su mejor desempeño. Los resultados destacaron a XGBoost como el mejor modelo. Se realizó una prueba t-Student que acepta la hipótesis alternativa en la mayoría de variables evaluadas, donde los datos predichos por XGBoost no presentaron diferencia significativa respecto a los datos reales. En conclusión, la implementación de un sistema informático inteligente con XGBoost logró predecir con mínimo margen de error las ventas diarias en la cevichería Rico Norte de la ciudad de Puno.

Palabras Clave: CRISP-DM, Machine Learning, Pronóstico de ventas, Sistema Informático Inteligente, XGBoost

ABSTRACT

The main objective of this thesis was to implement an intelligent information system to forecast daily sales at the cevicheria Rico Norte in Puno, Peru, as a solution to issues such as the lack and waste of supplies, delays in service during busy days, and the need to make business adjustments without harming its economy. The research was carried out in 2023 and 2024. The CRISP-DM development methodology, which consists of six phases, was used. In the business understanding phase, the needs and requirements were addressed. In the data understanding phase, the data provided by the owner was collected. In the data preparation phase, data preprocessing was performed. In the modeling phase, the SARIMA (Seasonal Autoregressive Integrated Moving Average), XGBoost (Extreme Gradient Boosting), LSTM (Long Short-Term Memory), and MLP (Multilayer Perceptron) Machine Learning models were implemented. In the evaluation phase, performance metrics such as MAE (Mean Absolute Error), MSE (Mean Squared Error), R^2 (Coefficient of Determination), and MAPE (Mean Absolute Percentage Error) were used. During the deployment phase, the system was implemented using HTML, CSS, JavaScript, Python, and SQL technologies, integrating the XGBoost model due to its superior performance. The results highlighted XGBoost as the best model. A t-Student test was conducted, accepting the alternative hypothesis for most of the evaluated variables, where the data predicted by XGBoost showed no significant difference from the real data. In conclusion, the implementation of an intelligent information system with XGBoost successfully predicted daily sales at Rico Norte cevichería in Puno with minimal error.

Keywords: CRISP-DM, Intelligent Information System, Machine Learning, Sales Forecasting, XGBoost

CAPÍTULO I

INTRODUCCIÓN

En los últimos años, nuestra creciente dependencia de la tecnología y la automatización hizo que éstas estén en prácticamente todos los aspectos de nuestras vidas.

Siendo una tendencia, la automatización cada vez coge más fuerza, puesto que estamos en una era en donde esta es esencial si queremos ser competitivos en nuestro entorno (Escala, 2024).

No podemos negar que el sector gastronómico no esté exento ante esta revolución tecnológica. Según Villaseñor (2024), la creciente apuesta a la automatización y la IA por parte del sector gastronómico está ayudando a los restaurantes a poder adaptarse a un mercado que cambia con rapidez y que se enfrenta a problemas con los suministros, escasez de mano de obra, inflación y clientes exigentes.

Actualmente, la Inteligencia Artificial dejó de ser un área de nicho para convertirse en una herramienta muy importante en varios sectores. Lenis (2024) señala que las empresas pueden usar a la IA para identificar patrones y tendencias ocultas en sus datos, permitiéndoles estar preparados para los cambios del mercado, mejorar sus estrategias de marketing y responder proactivamente a oportunidades o amenazas emergentes, dándoles una ventaja competitiva.

Mencionando a las predicciones, estas son muy importantes para la toma de decisiones en las empresas, y por qué no, también para los negocios. Andrade León (2022) menciona que, para una correcta toma de decisiones, las empresas deben realizar su planificación en base al uso de pronósticos.

La capacidad de prever lo que deparará el mercado mañana, en una semana, en un mes, o en un año, proporciona a los negocios una enorme ventaja competitiva. Esta capacidad de anticipar puede mejorar muchos aspectos del negocio, como ajustar la producción, planificar el inventario, proponer estrategias de marketing, entre otros.

En el contexto de la cevichería Rico Norte, la ausencia de un sistema informático limita su capacidad de poder lidiar con la actual demanda. Es por eso que la implementación de un sistema inteligente para el pronóstico de ventas no solo ofrece una oportunidad para modernizarse, sino también para abordar problemas importantes, como la gestión del inventario y la toma de decisiones, así como la predicción de ventas y la optimización de los pedidos.

1.1. DESCRIPCIÓN DEL PROBLEMA

La cevichería Rico Norte no es un establecimiento muy conocido en la ciudad de Puno, pero aún así tiene una generosa clientela, la cual puede ser impredecible en ciertos días. En esos días impredecibles, cuando hay demasiada clientela, los insumos preparados no llegan a ser suficientes, y la atención llega a ser deficiente, generando la molestia del comensal. En el caso de que haya poca clientela, los insumos preparados se desperdician, generando pérdidas para el negocio.

Por otro lado, la necesidad de planificar estrategias de marketing o realizar cambios en el negocio que no perjudiquen su economía también está presente.

Es por eso que se propuso implementar un sistema informático inteligente que sea capaz de suplir las necesidades y deficiencias de la cevichería Rico Norte de la ciudad de Puno.

1.2. FORMULACIÓN DEL PROBLEMA

¿Con qué precisión los modelos de machine learning logran predecir las ventas diarias de la Cevichería Rico Norte de la ciudad de Puno?

1.3. JUSTIFICACIÓN

En los últimos años, la digitalización en el Perú fue creciendo significativamente, en especial debido a la pandemia del COVID-19.

Según UTEC (2021), uno de los principales impactos de la pandemia fue el cambio de las dinámicas laborales y sus consecuencias económicas. Las reuniones presenciales de trabajo se convirtieron en videollamadas, muchos procesos fueron automatizados y la nube se volvió en el principal aliado.

Esta pandemia obligó a muchos negocios a adaptarse a la digitalización para no quebrar. Sin embargo, cuando la pandemia llegó a su fin, muchas pequeñas empresas en la ciudad de Puno, incluida la Cevichería Rico Norte, continuaron operando sin un sistema informático que les brinde eficiencia y mejore su administración.

Serrano Silvestre (2021) señala en los resultados de su investigación que, del 100% de los profesionales encuestados, el 49.1% consideran a la digitalización como necesaria para que la empresa pueda sobrevivir, y el 46.5% la consideran como muy relevante para la empresa. También señala que el 75.65% de profesionales encuestados consideran que la digitalización afectó de manera positiva en la rentabilidad de la empresa.

Por otro lado, según Romero (2022), el pronóstico de ventas juega un papel muy importante en las operaciones de cualquier negocio, ya que permite estimar la demanda, así como el flujo de caja.

Según el sitio web Datision (2021), con la llegada de la inteligencia artificial, las empresas tuvieron la capacidad de adelantarse a los acontecimientos. De esta forma, los grandes picos y valles de demanda no las toman desprevenidas, o no sufrirán de inconvenientes por falta de producción, entre otras ventajas. También el sitio web señala que la inteligencia artificial evitará que se tomen decisiones equivocadas que generarían sobrecoste, así como una parada de producción, la comercialización de un producto que no es demandado o que el mismo presente defectos. Entonces, para poder pronosticar las ventas, la inteligencia artificial, en especial el Machine Learning, se convertirá en una herramienta muy importante.

La Cevichería Rico Norte aún siguió usando métodos tradicionales para llevar a cabo su administración y demás operaciones diarias, lo que ha resultado en una disminución de la productividad a la hora de realizar y entregar los pedidos. Además, la incapacidad de pronosticar sus ventas ha llevado a una mala gestión de sus insumos, y a una deficiente toma decisiones. Esta situación generaba insatisfacción en los clientes, y resultaba en pérdidas económicas para el establecimiento.

El diseño y la implementación de un sistema informático inteligente para la Cevichería Rico Norte permitió pronosticar las ventas diarias mediante modelos de Machine Learning, y así mejorar la toma de decisiones mediante informes generados con los datos recolectados. Estos informes proporcionaron al dueño una vista detallada de las ventas, los gastos y los platos más demandados, entre otros aspectos relevantes. Asimismo, permitió anticipar la cantidad de clientes que acudían al establecimiento y qué platos fueron los más vendidos en fechas específicas.

Además, la implementación de este sistema informático inteligente pretendió mejorar la eficiencia en los procesos internos del negocio. Esto incluye la gestión de

pedidos y planificación de insumos, lo que puede llevar a una reducción de los tiempos de espera para los clientes y mejorar la gestión de los recursos, garantizando la rentabilidad del negocio.

La presente investigación no sólo beneficia a la Cevichería Rico Norte, asímismo puede servir como ejemplo y motivación para las pequeñas y medianas empresas de la ciudad de Puno, ya que la gran utilidad y beneficios que puede brindar un sistema informático inteligente fomentará la digitalización no sólo en el área gastronómica, sino también fortaleciendo el ecosistema empresarial en la ciudad de Puno y promoviendo su desarrollo económico.

El estudio de caso de la Cevichería Rico Norte en la ciudad de Puno también puede sentar las bases para futuros avances e investigaciones en el campo de la digitalización y el uso de la inteligencia artificial en el sector gastronómico.

1.4. OBJETIVOS

1.4.1. Objetivo General

Implementar un sistema informático inteligente para el pronóstico de ventas diarias de la Cevichería Rico Norte de la ciudad de Puno.

1.4.2. Objetivos Específicos

- Recopilar y preprocesar los datos de ventas proporcionados por el dueño de la cevichería rico norte, que alimentarán a los modelos de Machine Learning.
- Implementar los modelos de Machine Learning SARIMA, XGBoost, LSTM (Red Neuronal Recurrente) y MLP (Modelo de Perceptrón Multicapa).

- Evaluar cada modelo usando las métricas de Error Absoluto Medio (MAE), Error Cuadrático Medio (MSE, RMSE), R-cuadrado (R^2) y Error Porcentual Absoluto Medio (MAPE).
- Diseñar y codificar el sistema informático inteligente usando las tecnologías HTML, CSS, JavaScript, Python, Flask y SQL.

CAPÍTULO II

REVISIÓN DE LITERATURA

2.1. ANTECEDENTES DE INVESTIGACIÓN

Ponce Illacutipa (2022) desarrolló un modelo basado en Machine Learning para optimizar el pronóstico de ventas de la empresa Ricos Pan de la ciudad de Puno entre los años 2020 y 2021. En esta investigación se pusieron a prueba los siguientes modelos: Redes neuronales recurrentes (LSTM y GRU), Redes neuronales convolucionales (CNN) y Máquina de soporte de vectores (SVR). La investigación dio como conclusión que el modelo de redes convolucionales (CNN) obtuvo los mejores resultados. Dicho modelo superó significativamente al pronóstico manual que usaba la empresa.

Valverde Bourdié (2019) concluye que la inteligencia artificial no sólo facilita cualquier actividad dentro de la empresa, sino que hace evidente la eficiencia de éstas dentro del actual mercado, exigente y en constante cambio. Y de esta manera poder cumplir con las altas expectativas de los clientes.

Martel Vicente (2021) implementó un sistema web para el control de las ventas del comedor restaurante el CREVAL de la ciudad de Huánuco en el año 2020, con el objetivo de solucionar sus problemas de control y gestión de servicios. Se usó la metodología de Programación Extrema (XP). Los resultados obtenidos mostraron que el sistema obtuvo una usabilidad sobresaliente, evaluada mediante la Escala de Usabilidad del Sistema, teniendo alta aceptación por parte de los usuarios. Por otro lado, el sistema logró mejorar el control de ventas, el orden de pedidos y los reportes de ventas. Todo esto permitió una mejor gestión de las ventas del restaurante.

Saltos Intriago & Villacis Breilh (2022) implementaron modelos de Machine Learning en el área de ventas de la empresa Zapec S.A, concluyendo que la inteligencia artificial garantiza gran acierto en los resultados obtenidos, de esta forma comprobando su efectividad a comparación de los usos ocasionales de las empresas para organizar y hacer pronósticos a largo plazo, a la vez que se obtuvieron modelos más estables y confiables en la relación de variables.

Castañeda Rojas (2020) implementó modelos de Machine Learning en la gestión de ventas de la empresa Vértice Empresarial S.A.C., usando la metodología CRIPS-DM, con un tipo de investigación pre-experimental y enfoque cuantitativo. Se concluyó que el uso de modelos de Machine Learning aumentó el porcentaje de eficiencia de 54,64% a 82,04%, y el porcentaje de productividad de 51,54% a 81,07%. Afirmando así que el Machine Learning logró aumentar la eficiencia y productividad para la gestión de ventas en la empresa Vértice Empresarial S.A.C.

Nazmuz Sakib (2023) en su investigación Restaurant Sales Prediction Using Machine Learning concluye que la investigación proporcionó una descripción completa de los factores involucrados en la predicción de las ventas en restaurantes usando Machine Learning. Factores como el clima tienen un impacto significativo en las ventas y los pedidos de los restaurantes. También se identificó que la mayoría de los clientes prefieren pagar a través de servicios en línea. Durante los días festivos, en especial los domingos, hubo un gran número de pedidos y ventas. Además, se identificaron a quinientos clientes leales a los cuales se les puede brindar un mejor servicio y descuentos adicionales. La investigación destaca la importancia de la retroalimentación de los clientes para mejorar el rendimiento de los restaurantes. También recomienda implementar el análisis sentimental para una visualización efectiva en el análisis de datos.

Naveen Joshi (2023) en su investigación How Machine Learning Is Improving Restaurants concluye que el Machine Learning está transformando la industria de los restaurantes ya que les permite aprovechar el análisis de datos para mejoras las operaciones, la experiencia del cliente y la rentabilidad, pudiendo analizar grandes cantidades de datos para comprender las preferencias de los clientes, optimizar el inventario y desarrollar menús más atractivos para los clientes y, al mismo tiempo, rentables. El Machine Learning también permite detectar y prevenir fraudes en tiempo real, protegiendo sus negocios de posibles pérdidas.

Santiago et al. (2023) en su investigación Implementation of a Web System with Machine Learning for Sentiment Analysis in Social Networks for the Marketing tuvo como objetivo implementar un sistema web de Machine Learning para realizar un análisis de sentimientos en redes sociales para el área de marketing de la empresa D'Onofrio, todo esto con el fin de aumentar la efectividad de las estrategias de marketing y la toma de decisiones. La metodología usada combinó algoritmos de Machine Learning y procesamiento de lenguaje natural para analizar el contenido generado por los usuarios en plataformas de redes sociales. Los resultados obtenidos muestran que el análisis de emociones en redes sociales puede proporcionar valiosa información sobre las opiniones y emociones de los consumidores con respecto a los productos de D'Onofrio, proporcionando una comprensión más profunda sobre la percepción de la marca. Como conclusión, el estudio puso en evidencia la efectividad del análisis de sentimientos en redes sociales usando Machine Learning como una herramienta que puede mejorar la toma de decisiones en el ámbito del marketing de la empresa D'Onofrio.

Rohaan et al. (2022) en su investigación Using supervised machine learning for B2B sales forecasting: A case study of spare parts sales forecasting at an after-sales service provider presentó un método para poder utilizar información de demanda

anticipada (ADI), en forma de datos de solicitud de cotización (RFQ), para prever las ventas B2B. Se aplicaron técnicas de Machine Learning y procesamiento de lenguaje natural para el estudio de caso de un proveedor de servicios y mantenimiento postventa. En los resultados, las técnicas lograron identificar aproximadamente el 70% de las ventas reales con una precisión del 50%, lo que representa una mejora en comparación con el proceso manual que realizaba la empresa. Como conclusión, esta investigación proporciona una guía de cómo implementar inteligencia artificial para predecir las ventas B2B, indicando la mejora de rendimiento que se puede esperar al usar aprendizaje automático supervisado para este caso.

Yi (2023) en su investigación Walmart Sales Prediction Based on Machine Learning presenta un modelo de machine learning para predicción de ventas de Walmart combinado con un proceso de ingeniería de características. Se analizaron los modelos de regresión lineal, random forest, y XGBoost. Los resultados mostraron que el modelo XGBoost superó a los otros modelos mediante la métrica de evaluación (WAME). Como conclusión, esta investigación puede contribuir a una mejor comprensión del desarrollo de alternativas para la decisión en la industria minorista.

Pavlyshenko (2019) en su investigación Machine-learning models for sales time series forecasting estudió el uso de modelos de machine learning para el análisis de predicción de ventas. Se consideró el efecto de generalizar el aprendizaje automático, el cual se puede usar para predecir ventas de pequeñas cantidades de datos históricos en ventas específicas. Se estudió un enfoque de stacking para crear modelos de regresión individuales. Los resultados obtenidos muestran que el rendimiento de los modelos de predicción de ventas puede ser mejorado mediante las técnicas de apilamiento.

Saravanan et al. (2023) en su investigación Sales Prediction for Food Products Using Machine Learning identificó la venta total con mayor precisión, a la vez que se identifica la predicción de productos específicos, ventas por ubicación, ventas de supermercado, entre otras categorías. También se predijeron los rangos de ventas más altas con el uso de algoritmos KNN, Naive Bayes, entre otros.

Hussam Mezher Merdas & Ayad Hameed Mousa (2023) en su investigación Food Sales Prediction Using MLP, RANSAC, and Bagging propusieron un modelo que se divide en dos objetivos. La comparación de tres diferentes conjuntos de datos de ventas de alimentos, y la aplicación de tres algoritmos de inteligencia artificial para elegir el que proporciones la mayor precisión de predicción con el conjunto de datos especializado. Los algoritmos aplicados fueron Multilayer perceptrón, RANSAC, y regresión de Bagging, también se usaron diversas métricas para medir la precisión de los algoritmos. Los resultados obtenidos mostraron que el primer conjunto de datos obtuvo una mejor precisión con el algoritmo Bagging, a diferencia del segundo y tercer conjunto de datos, que obtuvieron bajas correlaciones. Como conclusión, el presente estudio sienta las bases para estudios posteriores y facilita la elección de un conjunto de datos.

Naik et al. (2022) en su investigación Machine Learning based Food Sales Prediction using Random Forest Regression tuvieron como objetivo encontrar los algoritmos más precisos para la predicción de ventas. Este estudio gira en torno a las técnicas de predicción y su ventajas y desventajas. Las técnicas más relevantes fueron Ingreasing Regression, Accidental Forestry Lapse y Random Forest. Los resultados obtenidos destacaron a Random Forest con un buen desempeño comparado con las otras técnicas de regresión, mejorando el accuracy score en 1.83% y reduciendo el error absoluto en 4.66%.

Schmidt et al. (2022) en su investigación Machine Learning Based Restaurant Sales Forecasting estudiaron varios modelos de machine learning usando datos de ventas reales de un restaurante. Se generaron tres conjuntos de datos para entrenar a los modelos y comparar resultados. Los resultados para predicción de un día fueron mejores con modelos lineales, los modelos RNN, LSTM y TFT también destacaron. Para la predicción semanal, solo destacó el modelo RNN. Como conclusión, los modelos de machine learning más simples tuvieron buen rendimiento al separar linealmente las instancias de entrenamiento.

Merdas & Mousa (2023) en su investigación Food sales prediction model using machine learning techniques propusieron un modelo de machine learning para la predicción de ventas de alimentos, el cual tuvo dos objetivos, los cuales fueron el de realizar una comparación entre dos conjuntos de datos, un conjunto con alta correlación, y otro con baja correlación; y utilizar varios algoritmos de machine learning para compararlos y encontrar los que brinden una mejor predicción. Mediante las métricas importantes, se obtuvo como resultado que la máquina de soporte de vectores (SVM), LASSO y Bagging fueron los mejores algoritmos para el primer conjunto de datos. Para el segundo conjunto de datos, los mejores algoritmos fueron Gradient Boosting, Random Forest y árboles de decisión (Decision Tree).

Mohanapriya & Mohana Saranya (2020) en su investigación Sales prediction using machine learning algorithm destacaron la precisión de predicción de ventas que se puede obtener usando algoritmos de machine learning. Concluyendo que XGBoost es mucho más rápido en su entrenamiento comparado con Gradient Boost, a la vez que Gradient Boost es más preciso para la predicción.

2.2. MARCO TEÓRICO

2.2.1. Sistema Informático

Un sistema informático (SI) es un conjunto de dispositivos, incluyendo al menos una CPU, que se conectan físicamente o de manera remota mediante software, permitiendo la interacción con agentes externos, como los seres humanos. Su objetivo es apoyar el procesamiento, almacenamiento, entrada y salida de datos, y se adapta con recursos específicos según su aplicación (Moreno Pérez & Ramos Pérez, 2014).

2.2.1.1. Elementos de un Sistema Informático

Según Moreno Pérez & Ramos Pérez (2014), el sistema de información debe contar con dos componentes esenciales: uno físico, conocido como hardware, y otro lógico, denominado software. Además de estos, es crucial considerar un tercer elemento, los recursos humanos, cuya presencia es indispensable para el funcionamiento, aunque no sea parte intrínseca del sistema en sí mismo. Asimismo, indica que tradicionalmente los elementos que componen un sistema informático son:

- Hardware: Son dispositivos físicos como terminales, canales de comunicación y medios de almacenamiento
- Software: Son programas informáticos para el procesamiento de datos, especialmente base de datos.
- Personal: Son usuarios y operadores del sistema.
- Documentación: Manuales y guías que detallan los procedimientos del sistema informático

2.2.2. Pronóstico de Ventas

El pronóstico de ventas viene a ser una técnica que nos permite proyectar las ventas de manera rápida y confiable, usando como datos las ventas realizadas o transacción de inventario. Otra finalidad del pronóstico de ventas es la de estimar la demanda a futuro, tomando de referencia el historial generado por las ventas o el movimiento de los productos (Toro Ocampo et al., 2004).

Clavijo (2023) agrega que un buen pronóstico de ventas permitirá a la empresa predecir las ventas alcanzables, gestionar eficazmente los recursos y poder diseñar un plan de crecimiento sostenido para el futuro. Dando a entender que lo anterior mencionado es el pilar de cualquier estrategia comercial, diseño de producto o planeamiento financiero.

También nos menciona que el pronóstico de ventas es la base para poder invertir recursos responsablemente, así como también definir objetivos reales a corto, medio y largo plazo.

2.2.3. Cross-Industry Standard Process (CRISP-DM)

Abbott (2014) define a CRISP-DM como el proceso que debe seguir un proyecto de minería de datos en seis pasos. Esto modelo fue reconocido como el más utilizado en este campo. Aunque algunas organizaciones utilizan sus propios modelos, estos siguen los mismos principios de CRISP-DM.

Una de las principales ventajas de CRISP-DM es su estructura, que es clara y bien documentada. A pesar de que este modelo está enfocado en la minería de datos, también se puede aplicar a campos como el análisis predictivo, estadística o analítica empresarial.

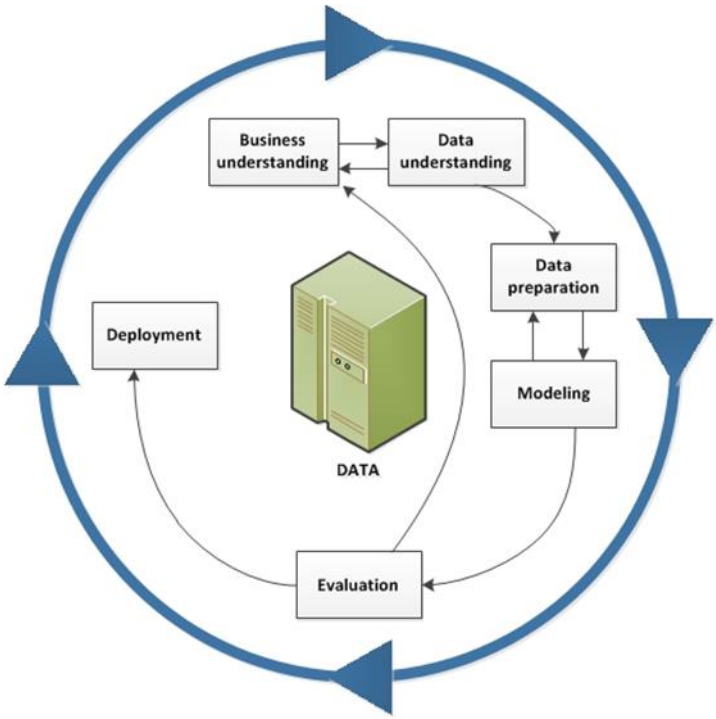
Por otra parte, IBM (2021a) define a CRISP-DM como una metodología y un modelo de proceso.

Se considera metodología porque describe las fases de un proyecto de minería de datos, especificando las tareas que cada una posee y explicando las relaciones entre ellas.

Se considera proceso porque ofrece una vista general sobre el ciclo de vida de un proyecto de minería de datos.

Figura 1

Ciclo vital de un proyecto de minería de datos.



Nota: Conceptos básicos de ayuda de CRISP-DM (IBM, 2021a).

Hotz (2024) describe de la siguiente manera las seis fases del modelo CRISP-DM:

2.2.3.1. Business Understanding (Compresión del Negocio)

Esta fase se enfoca en entender los requerimientos y objetivos del proyecto, donde se deben considerar cuatro tareas clave:

- Determinar los objetivos del negocio: Comprender lo que desea lograr el cliente desde su perspectiva empresarial, definiendo los criterios de éxito.
- Evaluar la situación: Determinar los requerimientos, recursos disponibles, evaluar riesgos y realizar análisis de costo-beneficio.
- Definir los objetivos: A parte de los objetivos, también se debe definir qué significa el éxito desde la perspectiva de la minería de datos.
- Elaborar el plan de proyecto: Seleccionar las herramientas y tecnologías, así como definir los planes detallados para cada fase del proyecto.

2.2.3.2. Data Understanding (Comprensión de los Datos)

Esta fase se enfoca en identificar, recopilar y analizar conjuntos de datos que serán necesarios para lograr los objetivos del proyecto:

- Recopilar datos iniciales: Obtener los datos necesarios y colocarlos en la herramienta de análisis.
- Describir los datos: Examinar y documentar los datos y sus propiedades.
- Explorar los datos: Profundizar en los datos visualizando o encontrando relaciones entre ellos.

- Verificar la calidad de los datos: Evaluar la limpieza de datos y documentar cualquier inconveniente.

2.2.3.3. Data Preparation (Preparación de datos)

También llamada "preprocesamiento de datos", esta fase prepara los conjuntos finales de datos para el modelado:

- Seleccionar datos: Determinar los datos que se usarán y documentar el porqué de su inclusión o exclusión.
- Limpieza de datos: Corregir, imputar o eliminar valores nulos. Sin la limpieza, se corre el riesgo de obtener resultados no deseados.
- Construir datos: Crear nuevas variables útiles a partir de las ya existentes.
- Integrar datos: Crear nuevos conjuntos de datos utilizando datos de otras fuentes.
- Formatear datos: Transformar los datos según la necesidad, como convertir valores en cadena a valores numéricos.

2.2.3.4. Modeling (Modelado)

Puede ser considerada la fase más emocionante de la ciencia de datos, y también la más corta. En esta fase se construyen y evalúan múltiples modelos y se divide en cuatro tareas:

- Seleccionar modelo: Determinar qué algoritmos son los más adecuados para lograr los objetivos (XGBoost, Regresión, Redes Neuronales, etc.).

- Generar diseño de pruebas: Según el enfoque del modelo, es necesario dividir los datos en conjuntos de entrenamiento, prueba y validación.
- Construir modelo: Implementar el algoritmo en un código, para posteriormente ejecutarlo.
- Evaluar modelo: Comparar múltiples modelos y evaluar su desempeño usando métricas de rendimiento.

2.2.3.5. Evaluation (Evaluación)

En esta fase se elige al modelo que mejor haya cumplido con los objetivos del negocio:

- Evaluar resultados: Evaluar si los modelos cumplen con los criterios de éxito del negocio, y cuáles se deberían aprobar.
- Revisar el proceso: Revisar el trabajo e identificar cualquier paso que se haya pasado por alto, posteriormente, resumir los hallazgos.
- Determinar los siguientes pasos: Decidir si se procede a la siguiente fase, se vuelven a realizar más pruebas o si se inician nuevos proyectos.

2.2.3.6. Deployment (Despliegue)

Dependiendo de los requisitos, esta fase puede ser tan simple como generar un informe, o tan compleja como implementar un proceso de minería de datos que se pueda repetir en toda la empresa.

Esta última fase tiene cuatro tareas:

- Planificar el despliegue: Documentar y desarrollar un plan para el despliegue del modelo.
- Planificar monitorización y mantenimiento: Desarrollar un plan de mantenimiento y monitoreo para evitar problemas al momento de que el modelo esté operativo.
- Informe final: Documentar un resumen sobre el proyecto.
- Revisar el proyecto: Ver el proyecto en retrospectiva para analizar qué funcionó bien y no tan bien, y cómo mejorar en el futuro.

2.2.4. Base de datos

En palabras de Elmasri & Navathe (2011), las bases de datos y la tecnología de bases de datos juegan un papel importante en casi todas las áreas donde se usan computadoras, esto incluye a negocios, E-commerce, ingeniería, derecho, educación, medicina, genética, etc.

Una base de datos es una colección de datos relacionados. Cuando hablamos de datos, nos referimos a hechos conocidos que se pueden registrar. Un ejemplo muy común son los nombres, direcciones y números de teléfono de las personas que conoces. Ya sea que las hayas registrado en una libreta o en un software como Access o Excel, esta colección es una base de datos.

Podríamos considerar que la colección de palabras de este párrafo es una base de datos, puesto que son datos relacionados, pero eso es un error. El uso del término de base de datos suele ser más restringido. Una base de datos posee las siguientes propiedades:

- Una base de datos representa algún aspecto del mundo real.

- Una base de datos es una colección de datos con algún significado inherente y lógicamente coherente.
- Una base de datos está diseñada, construida y poblada con datos para un propósito específico.

2.2.4.1. (Structured Query Language) SQL

AWS (2024d) define a SQL como un lenguaje de programación que almacena y procesa la información en una base de datos relacional. Este tipo de bases de datos tienen forma de tabla, con filas y columnas que representan los atributos y las relaciones entre los datos. En SQL se pueden usar instrucciones para almacenar, actualizar, eliminar, buscar o recuperar información de la base de datos, así como para mantener y optimizar su rendimiento.

2.2.5. Tecnologías de desarrollo de software

Según el sitio web StarTechUP (2021), las tecnologías de desarrollo de software, o también llamadas herramientas de generación de código, son programas informáticos usados para crear, mantener, modificar y depurar otros programas o aplicaciones.

Las tecnologías de software a continuación están orientadas en su mayoría al desarrollo web:

2.2.5.1. HTML

Para MDN (2023), el lenguaje de Marcas de Hipertexto o HyperText Markup Language (HTML), es considerado el componente más

básico de la web, puesto que se encarga de dar estructura y significado al contenido web.

La palabra “Hipertexto” está relacionada a los enlaces que vinculan páginas web entre sí.

Se utilizan “marcas” para poder etiquetar texto, imágenes u otro contenido que será mostrado en un navegador web.

2.2.5.2. CSS

Según MDN (2024a), el lenguaje de Hojas de Estilo en Cascada o Cascading Style Sheets (CSS) es usado para describir cómo deben presentarse los documentos HTML o XML. Este lenguaje describe la forma de renderizado del elemento estructurado en la pantalla, en el papel, en el habla o en otros medios.

2.2.5.3. JavaScript

Para MDN (2024b), JavaScript es un lenguaje de programación interpretado, que, si bien es usado en su mayoría para páginas web, también se puede usar en muchos otros entornos. JavaScript es un lenguaje basado en prototipos, multiparadigma, orientado a objetos, dinámico, imperativo y declarativo.

AWS (2024b) define a JavaScript como un lenguaje de programación usado para la creación de páginas web interactivas. Las funciones que puede realizar permiten mejorar la experiencia del usuario de un sitio web.

2.2.5.4. Python

En palabras de AWS (2024c), Python es un lenguaje de programación usado en desarrollo de software, aplicaciones web, ciencia de datos y Machine Learning. Al ser multiplataforma y fácil de aprender, es ampliamente usado por los desarrolladores para diferentes propósitos.

Matthes (2019) considera a Python como un lenguaje muy eficiente, porque permite crear programas con menos líneas de código y facilita escribir código limpio, en comparación con otros lenguajes. También destaca a la amplia comunidad que posee, dispuesta a aconsejar sobre el lenguaje y ayudar a la resolución de problemas.

2.2.5.5. Flask

Según el repositorio PyPi (2024), Flask es una librería o framework ligero de Python que permite crear aplicaciones web. Fue diseñado para su fácil y rápido uso, con la capacidad de poder escalar a aplicaciones más complejas.

Para Domingo Muñoz (2017), Flask facilita el desarrollo de aplicaciones web trabajando bajo el patrón de Modelo – Vista – Controlador (MVC). Flask contiene solo las herramientas esenciales para crear una aplicación web funcional, por eso se le considera “micro Framework”, pero puede complementarse con un gran conjunto de extensiones o plugins para aumentar su funcionalidad.

2.2.6. Herramientas de Inteligencia Artificial

Ortega (2024) define a las herramientas de inteligencia artificial como softwares inteligentes que usan algoritmos de machine learning y redes neuronales para imitar las funciones cognitivas humanas. Estas herramientas se usan en diferentes ámbitos, destacando los de investigación y empresariales, para optimizar operaciones, automatizar tareas, o descubrir valiosa información en grandes conjuntos de datos.

2.2.6.1. Tensorflow

Según Navarro (2024), TensorFlow es un framework de Python enfocado en la inteligencia artificial desarrollado por Google, que permite ejecutar operaciones matemáticas dentro de una CPU o GPU de manera optimizada.

González (2024) menciona que TensorFlow trabaja con “tensores”, los cuales forman parte de estructuras gráficas y nodos, y estos a su vez representan una operación matemática. Estos modelos gráficos permiten el despliegue de redes neuronales y, en consecuencia, la fácil implementación de algoritmos de Deep Learning.

2.2.6.2. ScikitLearn

González (2018) denomina a Scikit-Learn como la librería más útil de Python enfocada en Machine Learning, puesto que proporciona muchos algoritmos de aprendizaje supervisado y no supervisado, que van desde la regresión lineal, máquina de soporte de vectores (SVM), árboles de

decisión, hasta análisis factorial, análisis de componentes principales, redes neuronales no supervisadas, entre otros.

2.2.7. Preprocesamiento de Datos

Para García Fernández (2023), el preprocesamiento de datos es el conjunto de técnicas aplicadas a los datos antes de ser utilizados por los algoritmos de Machine Learning. Estas técnicas incluyen la limpieza, transformación y organización de los datos. De esta manera, los datos podrán ser adecuados para que los algoritmos puedan extraer información significativa y precisa, a la vez que ahorra tiempo y recursos de computación.

2.2.8. Machine Learning

El Machine Learning o Aprendizaje Automático es una forma de Inteligencia Artificial que permite a los sistemas aprender y mejorar su desempeño en tareas específicas sin ser necesariamente programados para dicho fin. Los algoritmos de Machine Learning son capaces de identificar patrones y predecir datos a través del análisis de los mismos. Muy diferentes a los sistemas programados de manera tradicional.

Según AWS (2022), el machine learning es la ciencia que desarrolla modelos y algoritmos estadísticos para ser usado en sistemas computacionales, con el objetivo de realizar tareas sin instrucciones explícitas, en lugar de basarse en patrones o inferencias.

Con una explicación más sencilla, IBM (2021b) señala que el machine learning es una rama de la inteligencia artificial y la informática que usa datos y

algoritmos para poder imitar el aprendizaje de los seres humanos, mejorando gradualmente su precisión.

Por otro lado, Mueller & Massaron (2019) definen al machine learning como una aplicación de la inteligencia artificial que tiene la capacidad de aprender y mejorar de forma automática, sin estar explícitamente programada para hacerlo. Este aprendizaje es el resultado de analizar cantidades de datos cada vez mayores.

El Machine Learning consiste en programar sistemas que optimicen un criterio de rendimiento haciendo uso de datos de ejemplo o experiencias pasadas. Teniendo un modelo definido con ciertos parámetros, el Machine Learning es el encargado de optimizar dichos parámetros usando los datos o experiencias pasadas que se entrenaron previamente (Alpaydin, 2014).

Con una explicación un poco más técnica, Alpaydin (2014) también agrega que el Machine Learning se basa en la teoría de la estadística para construir modelos matemáticos, haciendo inferencias a partir de una muestra. En la parte informática, se necesitan algoritmos eficientes para el entrenamiento que sean capaces de resolver problemas de optimización, como el de almacenar y procesar una gran cantidad de datos. Por otro lado, posterior al entrenamiento del modelo, su representación y solución logarítmica que permitan realizar la inferencia deben ser igual de eficientes.

El machine learning ofrece varias maneras de aprendizaje, dependiendo del tipo de dato y del resultado que se quiere obtener. Los estilos de aprendizaje son los siguientes:

Aprendizaje Supervisado:

Este aprendizaje trabaja con datos que están etiquetados y tienen un resultado esperado específico. Cuando el entrenamiento progresa, las predicciones son más precisas. Algunos algoritmos que entran en esta categoría son:

- Regresión Lineal o Logística
- Naive Bayes
- Máquina de Soporte de Vectores (SVM)
- K-Nearest Neighbors (KNN)

Aprendizaje No Supervisado:

Según IBM (2024), este tipo de aprendizaje usa algoritmos que analizan y agrupan dentro de clústeres datos sin etiquetar. Estos algoritmos pueden descubrir patrones ocultos sin necesidad de la intervención humana. Debido a esta capacidad, son eficientes para el análisis de datos exploratorios, el reconocimiento de imágenes, entre otros problemas. Algunos ejemplos de este tipo de aprendizaje son:

- Clustering o agrupamiento
- Redes Neuronales
- Detección de anomalías

Aprendizaje Auto-Supervisado:

En breves palabras de Mueller & Massaron (2019), el aprendizaje auto-supervisado brinda todos los beneficios del aprendizaje supervisado, pero sin la necesidad de etiquetar los datos. Este aprendizaje utiliza el contexto para realizar su tarea, y descubre por su cuenta las etiquetas de salida, pero no se puede usar

para estructurar datos. Su utilidad radica en resolver problemas que se experimentan al usar otros tipos de aprendizaje.

Aprendizaje por Refuerzo:

Para Sutton & Barto (2018), este aprendizaje consiste en aprender qué hacer y cómo actuar ante diversas situaciones, en lugar de recibir instrucciones sobre las acciones a realizar. El algoritmo debe descubrir qué acciones dan mayor recompensa mediante experimentación. En algunos casos, las acciones no solo afectan la recompensa inmediata, sino que también a las recompensas futuras.

2.2.8.1. Extreme Gradient Boosting (XGBoost)

Según AnalyticsVidhya (2024), XGBoost es un algoritmo de machine learning perteneciente al aprendizaje en conjunto. Hace uso de árboles de decisión como base para emplear técnicas de regularización que mejoran el modelo. XGBoost es popular debido a su eficiencia computacional, buen análisis y manejo de datos faltantes. Es usado mayormente para regresión, clasificación y ranking.

XGBoost es un método de aprendizaje en conjunto, este tipo de aprendizaje combina el poder predictivo de varios aprendices, dando como resultado un modelo único que ofrece la salida combinada de múltiples modelos.

Los aprendices más conocidos son Bagging y Boosting, ambos se pueden usar en varios modelos estadísticos, pero destacan en árboles de decisión.

Boosting:

AWS (2024a) describe al Boosting como un método de aprendizaje utilizado para reducir errores en la predicción mediante el entrenamiento secuencial de múltiples modelos para mejorar la precisión. Los modelos se dividen en aprendices débiles y aprendices fuertes. Por ejemplo, si se quiere entrenar un modelo que identifique gatos en imágenes, los aprendices débiles identificarán particularidades como los ojos, el color de pelaje o las orejas; estos aprendices se juntan para formar un aprendiz fuerte, que será capaz de identificar al gato y sus variaciones.

Continuando con el concepto de AnalyticsVidhya (2024), XGBoost se divide en tres pasos:

- Se define un modelo inicial denominado $\boldsymbol{F_0}$ que predecirá la variable $\boldsymbol{y}$ objetivo, y estará asociado con un residual $(\boldsymbol{y} - \boldsymbol{F_0})$.
- Un nuevo modelo $\boldsymbol{h_1}$ se ajusta a los residuales del primer paso.
- La combinación de $\boldsymbol{F_0}$ y $\boldsymbol{h_1}$ dan paso a $\boldsymbol{F_1}$, que viene a ser un modelo mejorado con un menor MSE (Error Cuadrático Medio) que el de $\boldsymbol{F_0}$.

$$F_1(x) \leftarrow F_0(x) + h_1(x)$$

Para mejorar $\boldsymbol{F_1}$, se puede crear otro modelo, llamado $\boldsymbol{F_2}$, a partir de los residuales de $\boldsymbol{F_1}$.

$$F_2(x) \leftarrow F_1(x) + h_2(x)$$

Se puede repetir este proceso $\boldsymbol{m}$ veces, hasta que el MSE se haya minimizado tanto como sea posible.

$$F_m(x) \leftarrow F_{m-1}(x) + h_m(x)$$

2.2.8.2. Red Neuronal Recurrente (RNN)

Una Red Neuronal Recurrente es un tipo de red neuronal que fue diseñada para trabajar con datos secuenciales, como series de tiempo, texto o secuencias biológicas. Se diferencia de las redes neuronales tradicionales debido a su capacidad de poder memorizar los estados anteriores, permitiéndole capturar la dependencia temporal en los datos.

Según AWS (2024e), una Red Neuronal Recurrente es un modelo de Machine Learning entrenado para convertir y procesar datos secuenciales a una secuencia de datos específica. Una RNN posee muchos componentes interconectados que son capaces de imitar las conversiones de datos que realizamos los humanos, como la traducción de un idioma.

Pasando a la explicación de Aggarwal (2018), en las secuencias, la entrada de datos tiene la forma de $\bar{\boldsymbol{x}}_1, \bar{\boldsymbol{x}}_2, \bar{\boldsymbol{x}}_3, \ldots, \bar{\boldsymbol{x}}_n$, donde cada $\bar{\boldsymbol{x}}_t$ es un punto que recibimos en el tiempo. Por ejemplo, $\bar{\boldsymbol{x}}_t$ puede contener valores medidos en el tiempo en una serie temporal.

Cada dato de una secuencia depende del dato anterior. A comparación de una red tradicional, las Redes Neuronales Recurrentes pueden procesar cada entrada al convertirlas a un estado oculto. Este estado oculto almacena información de las entradas anteriores,

actualizándose con cada nueva entrada, permitiendo a la RNN aprender y mantener el contexto de los datos a lo largo de la secuencia. Cada nueva entrada afecta al estado oculto y a la salida correspondiente.

El estado oculto está representado por la siguiente fórmula, que calcula el tiempo $\boldsymbol{t}$ usando la entrada en ese mismo tiempo y el estado oculto del tiempo anterior $\boldsymbol{t-1}$:

$$\bar{h}_t = f(\bar{h}_t - 1, \bar{x}_t)$$

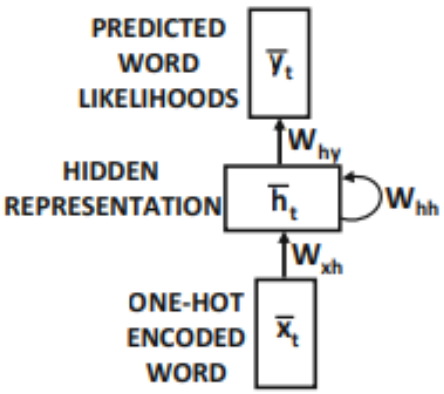
Para generar cada salida, se usa otra fórmula, que aprende las probabilidades a partir de los estados ocultos:

$$\bar{y}_t = g(\bar{h}_t)$$

El bucle mostrado en la Figura 2 cambia el estado oculto de la red neuronal después de cada entrada de $\bar{\boldsymbol{x}}_{\boldsymbol{t}}$.

Figura 2

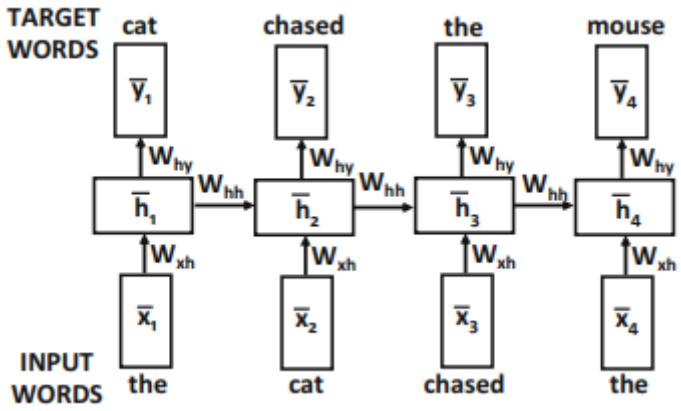
(a) Red Neuronal Recurrente.



Nota: Neural Networks and Deep Learning (Aggarwal, 2018).

Figura 3

(b) Representación temporal por capas de (a).



Nota: Neural Networks and Deep Learning (Aggarwal, 2018).

2.2.8.3. Long Short Term Memory (LSTM)

Según Graves (2012), LSTM consiste en subredes agrupadas en bloques, también llamadas bloques de memoria, que actúan como los chips de memoria de una computadora. Cada bloque contiene una o más celdas de memoria y tres puertas: la puerta de entrada, de salida, y de olvido. Dichas puertas tienen la función de escritura, lectura y reinicio para las celdas.

En lugar de sumar las activaciones en la capa oculta como lo hacía una Red Neuronal Recurrente (RNN), en LSTM las puertas ayudan a que las celdas de memoria puedan almacenar y acceder a información durante mucho tiempo, evitando así que el gradiente se desvanezca.

En palabras de Cheng et al. (2016), LSTM procesa una secuencia de datos, agregando de manera incremental nuevo contenido en una sola ranura de memoria. Las puertas controlan hasta qué punto se puede memorizar el nuevo contenido, borrar el contenido antiguo y exponer el

actual. Los valores requeridos para este procedimiento se expresan a continuación:

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ \hat{c}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W.[h_{t-1}, x_t]$$

$$c_t = f_t \square c_{t-1} + i_t \square \hat{c}_t$$

$$h_t = o_t \square \tanh(c_t)$$

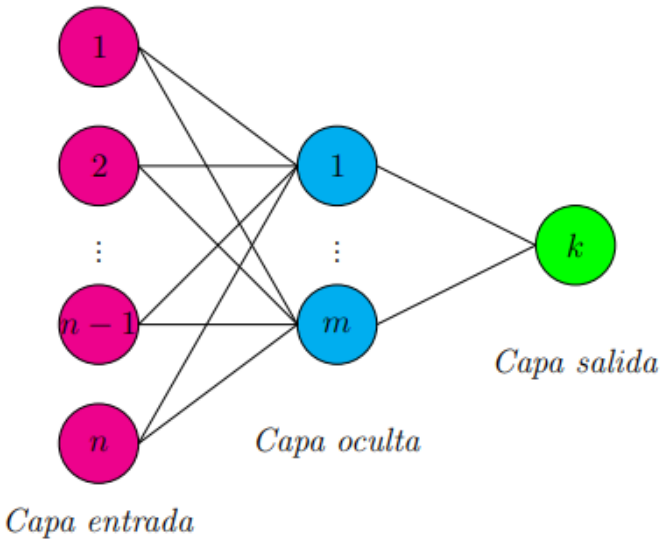
Donde $\boldsymbol{t}$ es el paso del tiempo, $\boldsymbol{c_t}$ la memoria, $\boldsymbol{h_t}$ el estado oculto, y como activadores de las puertas, $\boldsymbol{i}$, $\boldsymbol{f}$, y $\boldsymbol{o}$.

2.2.8.4. Modelo de Perceptrón Multicapa (MLP)

En palabras de Vivas (2014), el Modelo de Perceptrón Multicapa (MLP) consiste en una red neuronal unidireccional conformada por tres o más capas, una capa de entrada, otra de salida, y capas ocultas intermedias. La estructura de un perceptrón multicapa se puede apreciar en la Figura 4.

Figura 4

Estructura de Perceptrón Multicapa.



Nota: Optimización en el entrenamiento del Perceptrón Multicapa (Vivas, 2014).

González (2021) menciona que el Perceptrón Multicapa es una red de varias neuronas artificiales sobre capas variadas, y se activan de manera no lineal.

Este tipo de modelo es de las redes neuronales más sencillas, porque solo se componen de entradas, los pesos de cada entrada, la suma de la entrada ponderada, y la función de activación.

Entrada:

Las entradas están denotadas como $\boldsymbol{x_1, x_2, x_3, \ldots, x_n}$. Dichas entradas representan los valores de las características que recibirá el perceptrón.

Pesos:

Los pesos son valores que se ajustan durante el entrenamiento del perceptrón. Al principio se asignan aleatoriamente, y se actualizan cuando

ocurre un error en el entrenamiento. Están representados por $\boldsymbol{w_1, w_2, w_3, \ldots, w_n}$.

Suma ponderada:

La suma ponderada es la sumatoria de la multiplicación de cada valor de entrada por su peso correspondiente.

Función de activación:

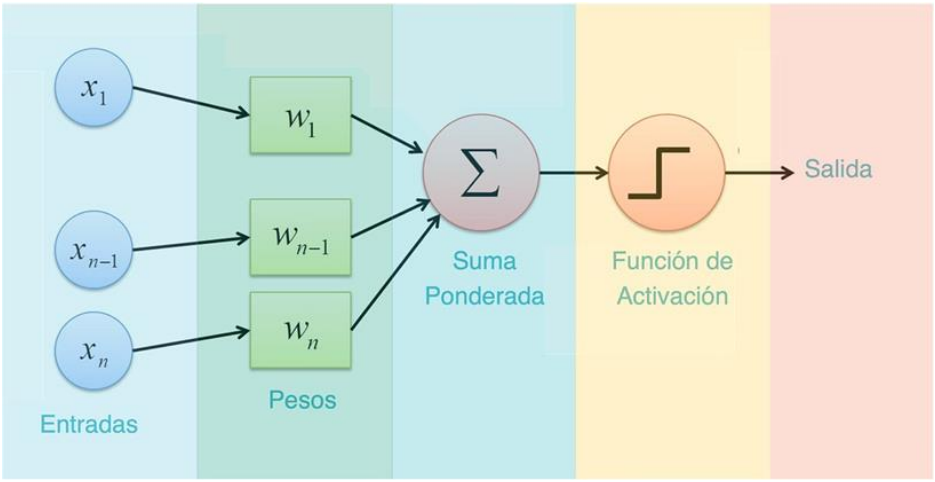
Esta función toma un solo número y realiza una ecuación específica sobre él. La función de activación suele ser Sigmoide o ReLU (Unidad Lineal Rectificada).

Salida:

El resultado de la suma ponderada se pasa a la función de activación, y el valor resultante es la salida obtenida por el perceptrón.

Figura 5

Representación de Modelo de Perceptrón.



Nota: ¿Qué es el Perceptrón? Perceptrón Simple y Multicapa (González, 2021).

2.2.9. Series de Tiempo

Según BookDown (2022), una serie de tiempo es una secuencia de datos que fueron medidos en un momento determinado, ordenados de forma cronológica y con intervalos iguales o desiguales.

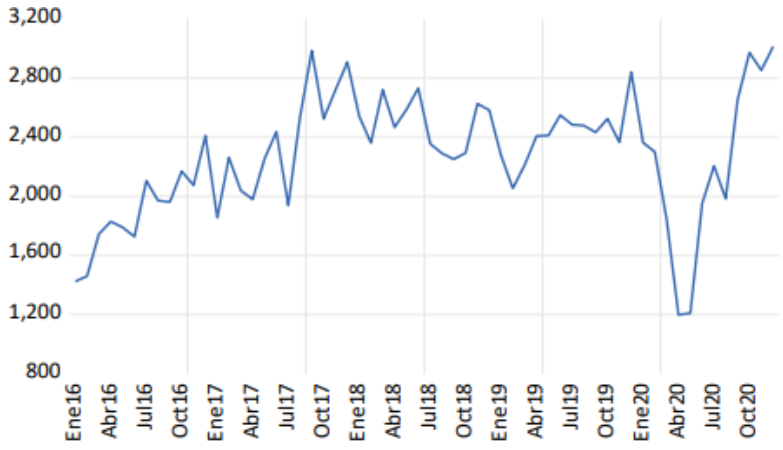
Analizar una serie de tiempo consiste en buscar un patrón o tendencia en los datos que la componen, pudiendo interpolar o extrapolar el comportamiento de nuevos datos basándose en los datos ya presentes.

Mills (2019) nos dice que las series de tiempo son omnipresentes y que aparecen en caso todas las investigaciones que analizan datos.

Una serie de tiempo de una variable se representa con x_t, donde t es el tiempo, que va desde 1 hasta T. A un periodo de tiempo completo se le denomina como periodo de observación. Estas observaciones se toman en intervalos regulares, como minutos, horas o días, es por eso que su orden es muy importante.

Figura 6

Modelo de pronóstico de una serie de tiempo de la exportación minera en el Perú desde enero 2016 hasta diciembre 2020.



Nota: Modelo de Pronóstico de la Exportación Minera en el Perú, 2020 (Izquierdo Henríquez et al., 2020).

2.2.9.1. SARIMA

Para explicar lo que es SARIMA, primero debemos tener en claro la definición de ARIMA.

El modelo autorregresivo integrado de media móvil o ARIMA, es una combinación de tres componentes: Autorregresión (AR), Diferenciación (I) y Media Móvil (MA).

González Casimiro (2009) señala que el modelo ARIMA se denota como:

$$ARIMA(p, d, q)$$

Donde $\boldsymbol{p}$ es el orden del polinomio autorregresivo estacionario, $\boldsymbol{d}$ el número de diferencias necesarias para estacionar la serie, y $\boldsymbol{q}$ la cantidad de términos de medias móviles invertible.

Para Chen et al. (2018), el modelo SARIMA, es un modelo ARIMA pero con términos estacionales añadidos. Este modelo se denota como:

$$ARIMA(p, d, q)(P, D, Q)m$$

Donde $(\boldsymbol{p}, \boldsymbol{d}, \boldsymbol{q})$ son las partes no estaciones vistas anteriormente, $(\boldsymbol{P}, \boldsymbol{D}, \boldsymbol{Q})$ las partes estacionales, y $\boldsymbol{m}$ el número de periodos por temporada.

La parte estacional es muy similar a la no estacional, pero se aplica a los intervalos estacionales.

2.2.10. Medición de rendimiento del modelo

Según DataBitAI (2023), la medición del rendimiento de un modelo se realiza mediante métricas de evaluación. Estas métricas comparan diferentes modelos y seleccionan al que tenga mejor rendimiento basándose en las necesidades del problema.

Existen múltiples métricas de evaluación, pero para los problemas de regresión se usan con más frecuencia el Error Cuadrático Medio (MSE), Error Absoluto Medio (MAE), Error Porcentual Absoluto Medio (MAPE), y el Coeficiente de Determinación (R2).

2.2.10.1. Error Cuadrático Medio (MSE)

SitioBigData (2018) señala al MSE como la métrica más simple, la cual mide el error cuadrado promedio de las predicciones. Calcula la diferencia al cuadrado de las predicciones y el objetivo en cada punto, para promediar los valores.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Donde $\boldsymbol{y_i}$ es el valor real esperado, $\boldsymbol{\hat{y}_i}$ la predicción, y $\boldsymbol{N}$ el número de observaciones.

Es útil si existen valores inesperados que deberían preocuparnos. Mientras más cercano a 0 sea el valor de MSE, el modelo será mejor.

2.2.10.2. Raíz del Error Cuadrático Medio (RMSE)

El RMSE es la raíz cuadrada del MSE. Se utiliza para poner a los errores en la misma escala que los objetivos, lo que facilita su interpretación.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} = \sqrt{MSE}$$

Donde $\boldsymbol{y_i}$ es el valor real esperado, $\boldsymbol{\hat{y}_i}$ la predicción, y $\boldsymbol{N}$ el número de observaciones.

2.2.10.3. Error Absoluto Medio (MAE)

El MAE es el promedio de las diferencias absolutas entre los valores objetivo y las predicciones. MAE se considera una puntuación lineal, donde todas las diferencias son ponderadas por igual en el promedio.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - \hat{y}_i\right|$$

Donde $\boldsymbol{y_i}$ es el valor real de la observación, $\boldsymbol{\hat{y}_i}$ el valor predicho de la observación, y $\boldsymbol{N}$ el número de observaciones.

La importancia del MAE radica en el hecho de que no es tan sensible a los valores atípicos, a diferencia del MSE.

2.2.10.4. Error Porcentual Absoluto Medio (MAPE)

El MAPE es una métrica que expresa los errores en términos porcentuales, facilitando su interpretación, pero es sensible a bajos volúmenes de datos o datos con periodos de 0 demanda.

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

Donde $\boldsymbol{y_i}$ es el valor real de la observación, $\hat{\boldsymbol{y}}_{\boldsymbol{i}}$ el valor predicho de la observación, y $\boldsymbol{N}$ el número de observaciones.

2.2.10.5.Coeficiente de Determinación ($\boldsymbol{R^2}$)

El $\boldsymbol{R^2}$ es una métrica relacionada con MSE, pero su única escala es estar entre $-\infty$ y 1. Representa la variabilidad total en la variable dependiente explicada por las variables independientes del modelo. Mientras más cercano sea el valor a 1, más preciso será el modelo.

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

Donde $\boldsymbol{y_i}$ es el valor real de la observación, $\hat{\boldsymbol{y}}_{\boldsymbol{i}}$ el valor predicho de la observación, $\bar{\boldsymbol{y}}$ el valor medio de los valores $\boldsymbol{y_i}$, y $\boldsymbol{N}$ el número de observaciones.

2.2.11. Prueba T-Student

Según Newbold et al. (2008), La distribución $\boldsymbol{t}$, desarrollada por Gosset, es la división de la distribución normal estándar entre la raíz cuadrada de la distribución chi-cuadrado dividida por sus grados de libertad.

Con $(\boldsymbol{n-1})$ grados de libertad, la variable $\boldsymbol{t}$ es definida como:

$$t = \frac{\bar{x} - \mu}{s / \sqrt{n}}$$

Donde $\boldsymbol{n}$ es el número de observaciones, $\bar{\boldsymbol{x}}$ la media de la muestra, $\boldsymbol{\mu}$ la media poblacional hipotética o conocida, y $\boldsymbol{s}$ la desviación estándar.

Es útil para muestra pequeñas, donde determina si existe diferencia significativa entre las medias de dos grupos.

2.3. HIPÓTESIS DE INVESTIGACIÓN

La implementación de un sistema informático inteligente basado en modelos de Machine Learning (XGBoost, SARIMA, LSTM y MLP) logrará predecir con un mínimo margen de error las ventas diarias de la Cevichería Rico Norte de la ciudad de Puno 2024.

2.4. VARIABLES

2.4.1. Variable dependiente

Modelo de pronóstico

2.4.2. Variable independiente

Rendimiento de pronóstico

2.5. OPERACIONALIZACIÓN DE VARIABLES

Tabla 1

Operacionalización de Variables.

VARIABLE	DIMENSIÓN	INDICADOR
Modelo de pronóstico	Método de pronóstico	- Modelo SARIMA - XGBoost - LSTM (Red Neuronal Recurrente) - MLP (Modelo de Perceptrón Multicapa)
Rendimiento de pronóstico	Medición de desempeño	- Error Absoluto Medio (MAE) - Error Cuadrático Medio (MSE, RMSE) - R-cuadrado (R^2) - Error Porcentual Absoluto Medio (MAPE)

CAPÍTULO III

MATERIALES Y MÉTODOS

3.1. DISEÑO Y TIPO DE INVESTIGACIÓN

3.1.1. Diseño de investigación

La presente investigación tuvo un diseño no experimental longitudinal. No experimental porque se manipulan los datos históricos mediante preprocesamiento para su correcto análisis, mas no para alterar su contexto natural. Y longitudinal porque los datos se basan en una serie temporal, la cual fue analizada mediante los modelos de pronóstico.

3.1.2. Tipo de investigación

La investigación aplicada se centra en estudiar un problema destinado a la acción. Este tipo de investigación concentra su atención en llevar a la práctica las teorías generales, aplicándose al mundo real, con el fin de satisfacer las necesidades que se plantea la sociedad (Baena Paz, 2017).

La presente investigación fue de tipo aplicada, porque se enfocó en desarrollar e implementar un sistema informático inteligente para el pronóstico de ventas diarias en la Cevichería Rico Norte, el cual buscó solucionar los problemas del negocio en un entorno de la vida real.

3.2. POBLACIÓN Y MUESTRA

3.2.1. Población

La población fue conformada por el número de ventas de los 60 productos ofrecidos por la cevichería Rico Norte durante los años 2022 y 2023.

3.2.2. Muestra

Se optó por usar una muestra no probabilística, puesto que no se realizó una selección aleatoria de los productos. Dicha muestra estuvo conformada por el número de ventas de los 25 productos más vendidos en la cevichería Rico Norte durante los años 2022 y 2023, puesto que estos representan en gran medida las ventas totales y tienen un impacto directo en los ingresos del negocio.

3.3. METODOLOGÍA DE DESARROLLO

Se usó la metodología CRISP-DM, que integra todas las tareas que se necesitan para este tipo de investigación.

Como es sabido, CRISP-DM se compone de 6 fases, las cuales son:

- Comprensión del negocio
- Comprensión de los datos
- Preparación de los datos
- Modelado
- Evaluación
- Despliegue

La fase de Comprensión del negocio abarcó las necesidades y requerimientos de la cevichería Rico Norte mediante una encuesta al dueño.

En la fase de Comprensión de datos se recopilaron los datos de venta proporcionados por el dueño de la cevichería Rico Norte.

En la fase de Preparación de datos, se realizaron la limpieza y preprocesamiento de los datos.

En la fase de Modelado se implementaron los modelos de Machine Learning, donde se probarán los siguientes modelos:

- Modelo SARIMA
- XGBoost
- LSTM (Red Neuronal Recurrente)
- MLP (Modelo de Perceptrón Multicapa)

En la fase de Evaluación se midió el rendimiento de los modelos mediante las siguientes métricas de desempeño:

- Error Absoluto Medio (MAE)
- Error Cuadrático Medio (MSE, RMSE)
- R-cuadrado (R2)
- Error Porcentual Absoluto Medio (MAPE)

En la fase de Despliegue se implementó el sistema informático inteligente desarrollado con las tecnologías web HTML, CSS, JavaScript, Python, Flask, SQL, y los modelos de predicción ya evaluados.

CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

4.1. APLICACIÓN DE LA METODOLOGÍA CRISP-DM

4.1.1. Comprensión del Negocio

Para poder abarcar los requerimientos y necesidades para la implementación del sistema informático inteligente, se le hizo al dueño una encuesta de requerimientos y necesidades para la implementación de un sistema informático inteligente para la Cevichería Rico Norte, de las cual se logró concluir que la cevichería tiene como objetivo principal mejorar su productividad y poder prever las ventas futuras.

La a veces incorrecta toma de los pedidos, entrega tardía de los mismos y el desperdicio o falta de insumos se destaca como un desafío actual.

Los datos se recopilan a mano sin una estructura definida, con la información de la fecha, las mesas y sus pedidos, y el precio total de la mesa.

Se espera que el sistema disminuya los errores al tomar los pedidos, prepare al personal prediciendo los días con más clientela, y evite el descontento de los clientes o el desperdicio de insumos con dicha predicción.

La expansión a futuro no fue considerada.

4.1.2. Comprensión de Datos

Los datos fueron proporcionados por el dueño de la cevichería Rico Norte, que fueron recolectados a mano entre los años 2022 – 2023.

Los datos recolectados a mano fueron transcritos a una tabla de Excel, la cual fue convertida a formato CSV.

La estructura de la tabla de Excel es la siguiente:

Tabla 2

Estructura de los Datos Históricos de ventas de la Cevichería Rico Norte en Excel.

Fecha	Total (S/.)	Producto_1	Producto_2	Producto_3	…	Producto_60
2022-01-01	2846	10	12	0		0
2022-01-03	1371	8	4	0		1
. . .						
2024-02-28	639	1	3	1		1

Nota: Elaboración Propia

Donde la columna Total almacena el monto diario obtenido, y cada columna de Producto almacena la cantidad diaria vendida de dicho producto.

Se eliminaron las columnas con productos que no tuvieron ni una sola venta, puesto que no serán relevantes para los procesos posteriores.

Los productos fueron codificados a producto_1, producto_2, producto_3, …, producto_60 para su fácil manejo.

La siguiente tabla muestra el nombre y el precio de cada producto:

Tabla 3

Productos de la Cevichería Rico Norte.

Producto	Nombre	Precio
producto_1	Ceviche Clásico	S/ 15.00
producto_2	Ceviche de Pescado	S/ 18.00
producto_3	Ceviche a la Crema, Macho, Olivo o Ají Amarillo	S/ 18.00
producto_4	Ceviche Mixto, Langostinos y Mariscos	S/ 28.00
producto_5	Ceviche de Pescado + Chicharrón de Pescado, Pota o Mixto	S/ 25.00
producto_6	Ceviche de Pescado + Chicharrón de Langostinos	S/ 30.00
producto_7	Ceviche Clásico + Chicharrón de Pescado, Pota o Mixto	S/ 24.00
producto_8	Ceviche Clásico + Chicharrón de Langostinos	S/ 28.00
producto_9	Ceviche Macho o Crema + Chicharrón de Pescado, Pota o Mixto	S/ 25.00
producto_10	Ceviche a lo Macho o Crema + Chicharrón de Langostinos	S/ 30.00
producto_11	Ceviche Tres Sabores + Chicharrón Mixto	S/ 35.00
producto_12	Tiradito Clásico, Macho, Crema, Olivo o Ají Amarillo	S/ 20.00
producto_13	Tiradito Clásico + Chaufa o Arroz con Mariscos y Langostinos	S/ 35.00
producto_14	Tiradito 3 Colores + Chaufa o Arroz con Mariscos y Langostinos	S/ 45.00
producto_15	Tiradito 5 Colores + Chaufa con Mariscos + Chicharrón Mixto	S/ 50.00
producto_16	Combinado Tradicional de 14	S/ 14.00
producto_17	Combinado Tradicional de 18	S/ 18.00
producto_18	Combinado con Mariscos	S/ 24.00
producto_19	Combinado con Langostinos	S/ 26.00
producto_20	Combinado con Mariscos y Langostinos	S/ 28.00
producto_21	Triple Tradicional	S/ 23.00
producto_22	Triple a la Crema o Macho	S/ 24.00
producto_23	Triple con Mariscos a lo Macho o Crema	S/ 28.00
producto_24	Triple (1)	S/ 28.00
producto_25	Triple (2)	S/ 30.00
producto_26	Dúo Clásico	S/ 20.00
producto_27	Ronda Marina (1)	S/ 40.00
producto_28	Ronda Marina (2)	S/ 45.00
producto_29	Chicharrón de Pescado-Porción	S/ 13.00
producto_30	Chicharrón de Pescado-Entero	S/ 25.00

Producto	Nombre	Precio
producto_31	Chicharrón de Pota-Porción	S/ 12.00
producto_32	Chicharrón de Pota-Entero	S/ 23.00
producto_33	Chicharrón Mixto de Pescado y Pota	S/ 25.00
producto_34	Chicharrón Mixto (Pescado, Langostino y Pota)	S/ 30.00
producto_35	Chicharrón de Langostinos	S/ 30.00
producto_36	Arroz con Mariscos + Chicharrón de Pescado, Pota o Mixto	S/ 26.00
producto_37	Chaufa Tradicional + Chicharrón de Pescado o Mixto	S/ 17.00
producto_38	Chaufa Tradicional + Chicharrón de Pota	S/ 15.00
producto_39	Chaufa con Mariscos y Langostinos	S/ 28.00
producto_40	Arroz con Mariscos y Langostinos	S/ 30.00
producto_41	Causa Acevichada	S/ 22.00
producto_42	Chilcano Acevichado	S/ 10.00
producto_43	Cevichop	S/ 22.00
producto_44	Leche de Tigre (Crema, Macho, Olivo, Ají Amarillo)	S/ 15.00
producto_45	Leche de Tigre de 5	S/ 5.00
producto_46	Leche de Tigre de 12	S/ 12.00
producto_47	Arroz Chaufa (Porción)	S/ 5.00
producto_48	Jarra de Chicha	S/ 8.00
producto_49	Gaseosa 1 Litro	S/ 8.00
producto_50	Gaseosa 2 Litros y medio	S/ 12.00
producto_51	Gaseosa 1 Litro y medio	S/ 10.00
producto_52	Gaseosa medio Litro	S/ 4.00
producto_53	Gaseosa Gordita	S/ 5.00
producto_54	Gaseosa Personal	S/ 2.50
producto_55	Chicha 2 Litros	S/ 6.00
producto_56	Chicha Personal	S/ 2.00
producto_57	Cerveza Cuzqueña	S/ 10.00
producto_58	Cerveza Pilsen	S/ 9.00
producto_59	Combinado a la Crema o a lo Macho	S/ 18.00
producto_60	Agua Mineral	S/ 2.00

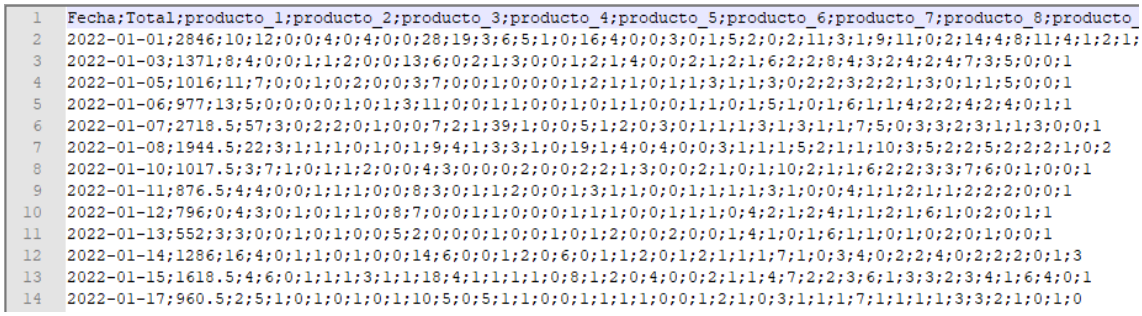
Nota: Productos que figuran en la Carta de la Cevichería Rico Norte

4.1.3. Preparación de Datos

Los datos transcritos a la tabla Excel fueron trasformados al formato de archivo CSV delimitado por puntos y comas (";"), como se muestra a continuación:

Figura 7

Datos de ventas diarias de la Cevichería Rico Norte en formato CSV.



```
Fecha;Total;producto_1;producto_2;producto_3;producto_4;producto_5;producto_6;producto_7;producto_8;producto_
2022-01-01;2846;10;12;0;0;4;0;4;0;0;28;19;3;6;5;1;0;16;4;0;0;3;0;1;5;2;0;2;11;3;1;9;11;0;2;14;4;8;11;4;1;2;1;
2022-01-03;1371;8;4;0;0;1;1;2;0;0;13;6;0;2;1;3;0;0;1;2;1;4;0;0;2;1;2;1;6;2;2;8;4;3;2;4;2;4;7;3;5;0;0;1
2022-01-05;1016;11;7;0;0;1;0;2;0;0;3;7;0;0;1;0;0;0;1;2;1;1;0;1;1;3;1;1;3;0;2;2;3;2;2;1;3;0;1;1;5;0;0;1
2022-01-06;977;13;5;0;0;0;0;1;0;1;3;11;0;0;1;1;0;0;1;0;0;1;0;1;1;0;0;1;1;0;1;5;1;0;1;6;1;1;4;2;2;4;2;4;0;1;1
2022-01-07;2718.5;57;3;0;2;2;0;1;0;0;7;2;1;39;1;0;0;5;1;2;0;3;0;1;1;1;3;1;3;1;1;1;7;5;0;3;3;2;3;1;1;3;0;0;1
2022-01-08;1944.5;22;3;1;1;1;0;1;0;1;9;4;1;3;3;1;0;19;1;4;0;4;0;0;3;1;1;1;5;2;1;1;10;3;5;2;2;5;2;2;2;1;0;2
2022-01-10;1017.5;3;7;1;0;1;1;2;0;0;4;3;0;0;0;2;0;0;2;2;1;3;0;0;2;1;0;1;10;2;1;1;6;2;2;3;3;7;6;0;1;0;0;1
2022-01-11;876.5;4;4;0;0;1;1;1;0;0;8;3;0;1;1;2;0;0;1;3;1;1;0;0;1;1;1;1;3;1;0;0;4;1;1;2;1;1;2;2;2;0;0;1
2022-01-12;796;0;4;3;0;1;0;1;1;0;8;7;0;0;1;1;0;0;0;1;1;1;0;0;1;1;1;0;4;2;1;2;4;1;1;2;1;6;1;0;2;0;1;1
2022-01-13;552;3;3;0;0;1;0;1;0;0;5;2;0;0;0;1;0;0;1;0;1;2;0;0;2;0;0;1;4;1;0;1;6;1;1;0;1;0;2;0;1;0;0;1
2022-01-14;1286;16;4;0;1;1;0;1;0;0;14;6;0;0;1;2;0;6;0;1;1;2;0;1;2;1;1;1;7;1;0;3;4;0;2;2;4;0;2;2;2;0;1;3
2022-01-15;1618.5;4;6;0;1;1;1;3;1;1;18;4;1;1;1;1;0;8;1;2;0;4;0;0;2;1;1;4;7;2;2;3;6;1;3;3;2;3;4;1;6;4;0;1
2022-01-17;960.5;2;5;1;0;1;0;1;0;1;10;5;0;5;1;1;0;0;1;1;1;1;0;0;1;2;1;0;3;1;1;1;7;1;1;1;1;3;3;2;1;0;1;0
```

Nota: Elaboración propia mediante Notepad++

El archivo CSV con nombre ventasCev.csv se cargó a Python usando **read_csv** que proporciona la librería *pandas*, se cambió el formato de la fecha a objeto *datetime* para evitar errores, y se creó una variable para poder explorar cada columna del *dataframe* con más facilidad.

```
df=pd.read_csv('C:/Users/Rodrigo/Desktop/SISTEMA_CEVICHERIA/ventasCev.
csv', delimiter=';')

df['Fecha'] = pd.to_datetime(df['Fecha'], format='%Y-%m-%d')

variable= "Total"
```

Mediante el archivo CSV cargado a Python, se generó una descripción inicial de los datos.

La función *describe* da una visión general de la media, cuartiles, datos mínimos y máximos, y la desviación estándar de cada columna.

La función *info* indica el tipo de dato de cada columna, y si existe algún dato faltante.

```
print("Descripción de los datos:")
print(df.describe())
print("\nInformación de los datos:")
print(df.info())
```

Figura 8

Descripción de los datos de venta de la Cevichería Rico Norte.

```
Descripción de los datos:
                                Fecha        Total  ...  producto_59  producto_6
0
count                             626   626.000000  ...   626.000000   626.00000
0
mean   2023-02-03 01:20:30.670926592  1241.110224  ...     0.428115     0.80670
9
min              2022-01-01 00:00:00    67.000000  ...     0.000000     0.00000
0
25%              2022-07-18 06:00:00   636.125000  ...     0.000000     0.00000
0
50%              2023-02-10 12:00:00  1016.750000  ...     0.000000     1.00000
0
75%              2023-08-23 18:00:00  1658.250000  ...     1.000000     1.00000
0
max              2024-02-28 00:00:00  5029.500000  ...     8.000000     7.00000
0
std                              NaN   817.338942  ...     0.600353     0.89586
8

[8 rows x 45 columns]
```

Nota: Elaboración propia

Figura 9

Información de los datos de venta de la Cevichería Rico Norte.

```
Información de los datos:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 626 entries, 0 to 625
Data columns (total 45 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Fecha        626 non-null    datetime64[ns]
 1   Total        626 non-null    float64
 2   producto_1   626 non-null    int64
 3   producto_2   626 non-null    int64
 4   producto_3   626 non-null    int64
 5   producto_4   626 non-null    int64
 6   producto_5   626 non-null    int64
 7   producto_6   626 non-null    int64
 8   producto_7   626 non-null    int64
 9   producto_8   626 non-null    int64
 10  producto_9   626 non-null    int64
 11  producto_16  626 non-null    int64
 12  producto_17  626 non-null    int64
 13  producto_18  626 non-null    int64
 14  producto_21  626 non-null    int64
```

Nota: Elaboración propia

La función isnull determina si existen valores nulos en las columnas.

```
print("\nComprobación de valores nulos:")
valores_nulos = df.isnull().sum()

if valores_nulos.sum() == 0:
    print("No se encontraron valores nulos.")
else:
    print(valores_nulos)
```

El script no encontró valores nulos.

Se procedió a modificar el dataframe para solo considerar los 25 productos con mayor cantidad de ventas.

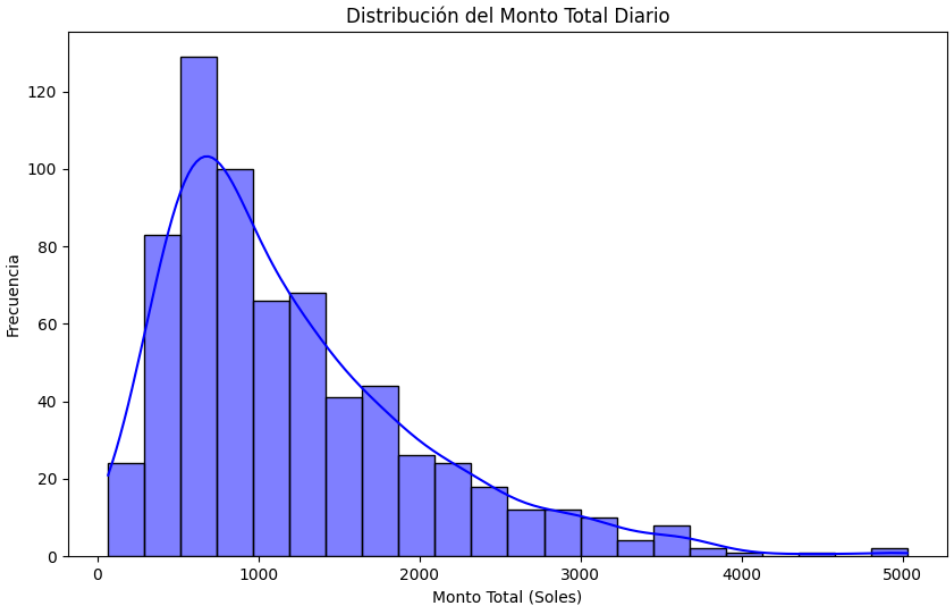
```
suma_ventas= df.iloc[:, 3:].sum()
mas_vendidos= suma_ventas.nlargest(25)
df= df[['Fecha', 'Total'] + mas_vendidos.index.tolist()]
```

Se realizó una distribución del monto total diario usando las librerías *Matplotlib* y *Seaborn*.

```
plt.figure(figsize=(10, 6))
sns.histplot(df['Total'], kde=True, color='blue')
plt.title('Distribución del Monto Total Diario')
plt.xlabel('Monto Total (Soles)')
plt.ylabel('Frecuencia')
plt.show()
```

Figura 10

Distribución del Monto Total Diario de las ventas de la Cevichería Rico Norte.



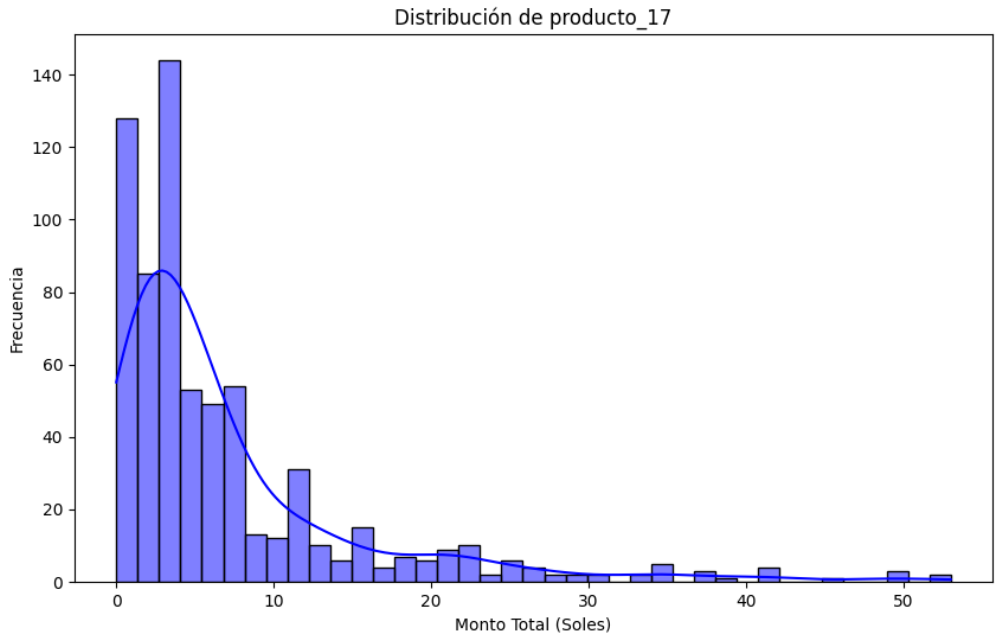
Nota: Elaboración propia

En la Figura 10 se puede visualizar que la Cevichería Rico Norte genera un monto que oscila entre los 800 y 1000 soles en la mayoría de los días.

Se puede realizar la distribución de cualquiera de los 25 productos, en este caso, se realizó la distribución del *producto_17*.

Figura 11

Distribución de ventas diarias del producto_17.



Nota: Elaboración propia

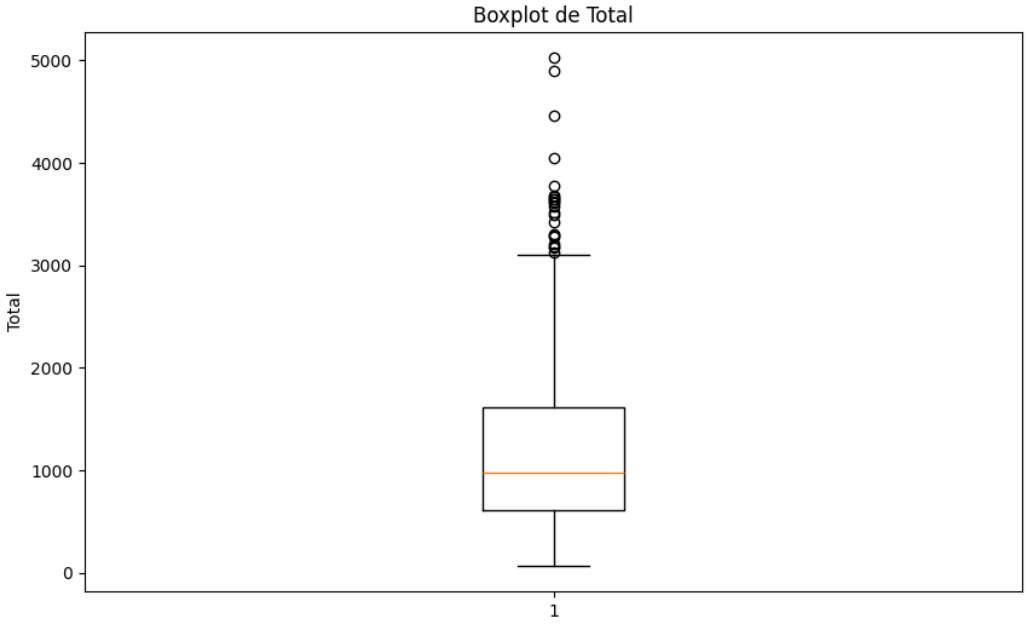
En la Figura 11 se aprecia que se venden con más frecuencia de 0 a 9 unidades del producto_17 (Combinado Tradicional de 18) al día.

Para poder identificar valores atípicos (outliers), se realizó un gráfico de caja (box plot).

```
plt.figure(figsize=(10, 6))
plt.boxplot(df[variable])
plt.title('Boxplot de '+ variable)
plt.ylabel('Total')
plt.show()
```

Figura 12

Boxplot del Total de ventas diarias de la Cevichería Rico Norte.



Nota: Elaboración propia

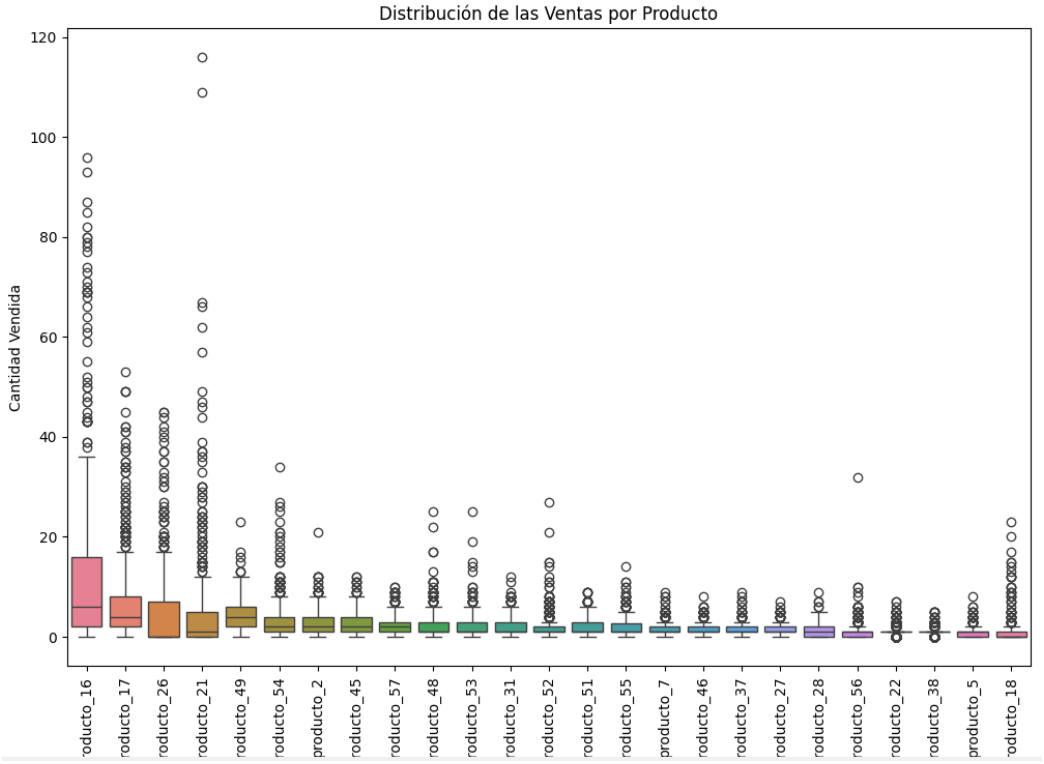
Como se ve en la Figura 12, se pueden apreciar valores atípicos de la columna Total que van desde los 3000 hasta los 5000 soles diarios.

También se puede apreciar los Boxplot de las demás columnas.

```
productos_cols = df.columns[2:]
print("\nDescripción de las columnas de productos:")
print(df[productos_cols].describe())

plt.figure(figsize=(12, 8))
sns.boxplot(data=df[productos_cols])
plt.title('Distribución de las Ventas por Producto')
plt.xlabel('Productos')
plt.ylabel('Cantidad Vendida')
plt.xticks(rotation=90)
plt.show()
```

Figura 13

Boxplot de las ventas diarias de los productos de la Cevichería Rico Norte.



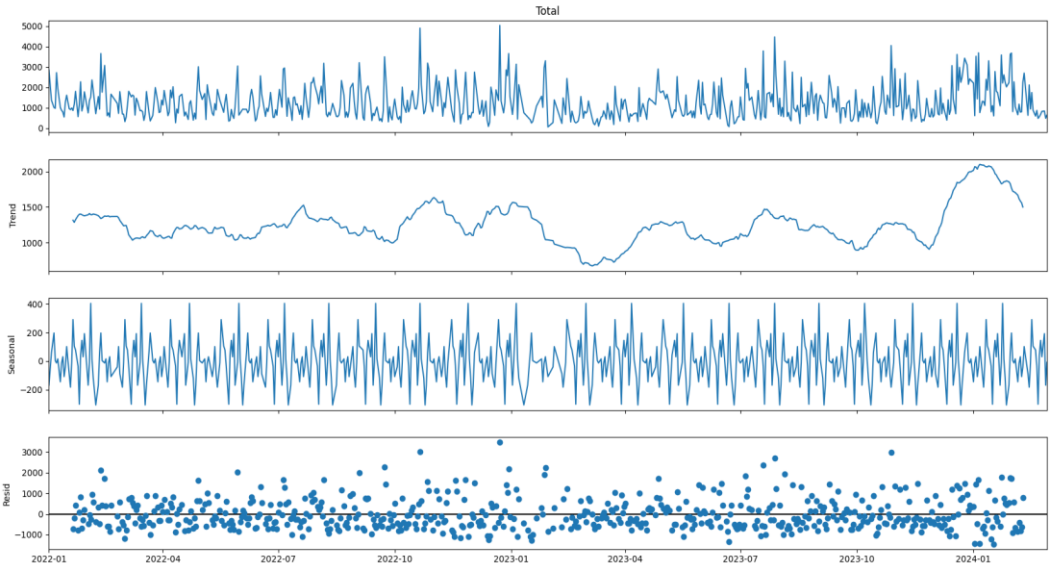
Nota: Elaboración propia

Antes de decidir si se eliminan o se mantienen los valores atípicos, se debe analizar la tendencia de los datos a través del tiempo mediante la descomposición, en este caso, de la variable Total, para ello se hizo uso de la librería Statsmodels, y se definió como índice a la columna Fecha.

```
df.set_index('Fecha', inplace=True)
result = seasonal_decompose(df[variable], model='additive', period=30)
result.plot()
plt.show()
```

Figura 14

Descomposición del Total de ventas diarias de la Cevichería Rico Norte.



Nota: Elaboración propia

La Figura 14 muestra una serie temporal como primer gráfico, seguido de una tendencia, una estacionalidad y un residual.

Los gráficos más relevantes que parecen explicar el porqué de los valores atípicos son la estacionalidad y el residual. En la estacionalidad se puede apreciar un patrón muy repetitivo, dando a entender que hay picos de ventas en cada mes. En el residual se observa cierta dispersión, pero no aparenta estar de manera aleatoria.

Es probable que los valores atípicos estén relacionados a eventos estacionales. Mantenerlos puede ayudar a mejorar los modelos para que sean capaces de predecir estos eventos. Para poder apreciar si los *outliers* benefician o empeoran el modelo, se entrenaron los modelos con *outliers* y sin *outliers*.

Para mejorar el rendimiento de los modelos, antes de realizar el entrenamiento se añadieron al *dataframe* características temporales basándose en los datos actuales.

Las nuevas características derivadas de la columna *Fecha* permiten identificar patrones y estacionalidades con mayor facilidad.

```
#Características adicionales
df['Dia_de_semana'] = df['Fecha'].dt.dayofweek
df['Dia_del_mes'] = df['Fecha'].dt.day
df['Dia_del_año'] = df['Fecha'].dt.dayofyear
df['Año'] = df['Fecha'].dt.year
df['Mes'] = df['Fecha'].dt.month
df['Es_fin_de_semana'] = df['Dia_de_semana'].isin([5, 6]).astype(int)
df['Semana_del_año'] = df['Fecha'].dt.isocalendar().week
df['Trimestre'] = df['Fecha'].dt.quarter
```

Las nuevas características derivadas de las columnas *Total, producto_16, producto_17*, etc., permiten identificar si las ventas de un día anterior influyen en el día siguiente, o si las ventas de una semana anterior influyen en la semana siguiente.

```
df['Lag_1'] = df[variable].shift(1)
df['Lag_7'] = df[variable].shift(7)
```

Las medias móviles derivadas de la columna Total, producto_16, producto_17, etc., permiten suavizar la serie temporal para detectar tendencias a corto y largo plazo.

```
#Medias móviles
df['Media_movil_3'] = df[variable].rolling(window=3).mean()
df['Media_movil_7'] = df[variable].rolling(window=7).mean()
df['Media_movil_30'] = df[variable].rolling(window=30).mean()
df['Diferencia_dia_anterior'] = df[variable].diff()
```

4.1.4. Modelado

Después de haber preprocesado los datos, se pusieron a prueba los modelos de Machine Learning, los cuales fueron:

- Modelo SARIMA
- XGBoost
- LSTM (Red Neuronal Recurrente)
- MLP (Modelo de Perceptrón Multicapa)

Se codificó un script de Python para cada modelo, empezando por el Modelo SARIMA.

SARIMA:

Para poder realizar el entrenamiento, se usaron como apoyo las librerías Statsmodels y Scikitlearn.

Para la prueba se usó a la variable que representa el Total de ventas diarias de la Cevichería Rico Norte.

```
df=pd.read_csv('C:/Users/Rodrigo/Desktop/SISTEMA_CEVICHERIA/ventasCev_
modificado.csv', delimiter=',')
df['Fecha'] = pd.to_datetime(df['Fecha'], format='%Y-%m-%d')
df.set_index('Fecha', inplace=True)
print(df.head())

variable= "Total"
```

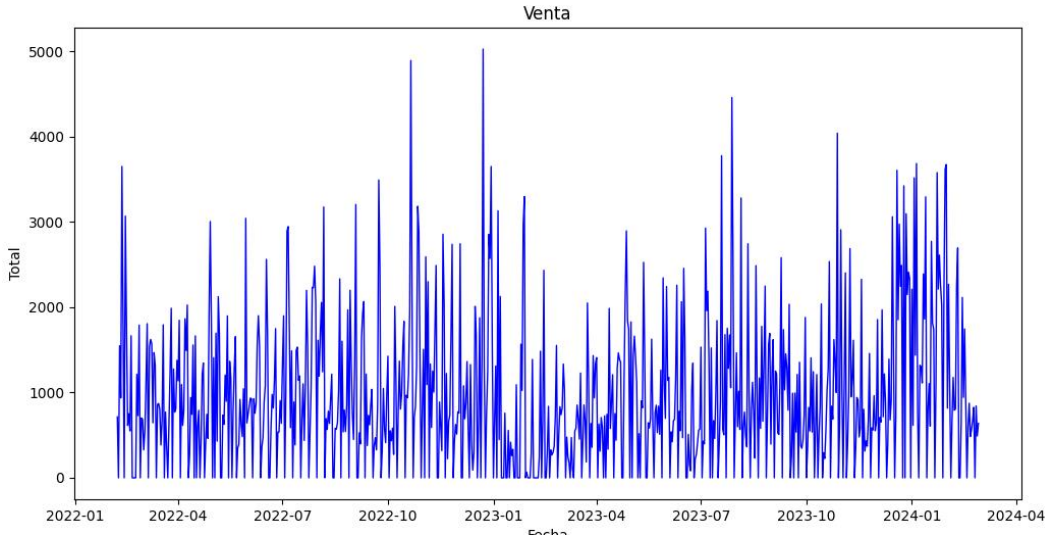
Primero se procedió sin eliminar los valores atípicos. Se definió la variable dependiente X.

```
X = df[[variable, 'Dia_de_semana', 'Dia_del_mes', 'Dia_del_año',
'Año', 'Mes', 'Es_fin_de_semana', 'Semana_del_año', 'Trimestre',
'Lag_1', 'Lag_7', 'Media_movil_3', 'Media_movil_7', 'Media_movil_30',
'Diferencia_dia_anterior']].resample('D').sum()
```

Se graficó la serie temporal del Total de ventas diarias de la Cevichería Rico Norte.

Figura 15

Serie Temporal del Total de ventas diarias de la Cevichería Rico Norte.



Nota: Elaboración propia

Se sabe que SARIMA se compone de $(\boldsymbol{p}, \boldsymbol{d}, \boldsymbol{q})$ y $(\boldsymbol{P}, \boldsymbol{D}, \boldsymbol{Q})\boldsymbol{m}$. Donde $\boldsymbol{p}$ es el orden del polinomio autorregresivo estacionario, $\boldsymbol{d}$ el número de diferencias necesarias para estacionar la serie, y $\boldsymbol{q}$ la cantidad de términos de medias móviles invertible. $(\boldsymbol{P}, \boldsymbol{D}, \boldsymbol{Q})$ las partes estacionales, y $\boldsymbol{m}$ el número de periodos por temporada.

Para calcular $\boldsymbol{d}$, se usó el test de *Dickey-Fuller* proporcionado por la función **adfuller**.

```
def estacionariedad(timeseries):
    #Dickey-Fuller test
    result = adfuller(timeseries, autolag='AIC')
    p_value = result[1] #El p-valor obtenido rechaza la H0 y concluye
que la serie es estacionaria, por ende d = 0
    print(f'ADF Statistic: {result[0]}')
    print(f'p-value: {p_value}')
    print('Estacionaria' if p_value < 0.05 else 'No Estacionaria')
```

```
estacionariedad(X[variable])
```

Figura 16

Estacionariedad de la serie de tiempo del Total de ventas diarias de la Cevichería Rico Norte.

```
ADF Statistic: -4.026117575440161
p-value: 0.0012797602184745347
Estacionaria
```

Nota: Elaboración propia

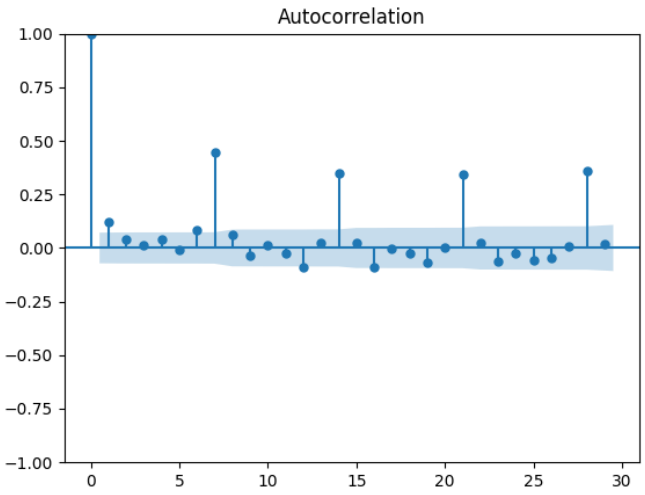
El p-valor menor a 0.05 indica que se acepta la Hipótesis alternativa y, por lo tanto, la serie es estacionaria sin necesidad de iterar. Como conclusión, el valor $\boldsymbol{d}$ es igual a 0.

Para calcular $\boldsymbol{p}$ y $\boldsymbol{q}$, se generó un gráfico de Función de Autocorrelación (ACF) y un gráfico de Función de Autocorrelación Parcial (PACF).

```
plot_acf(X[variable])
plot_pacf(X[variable])
plt.show()
```

Figura 17

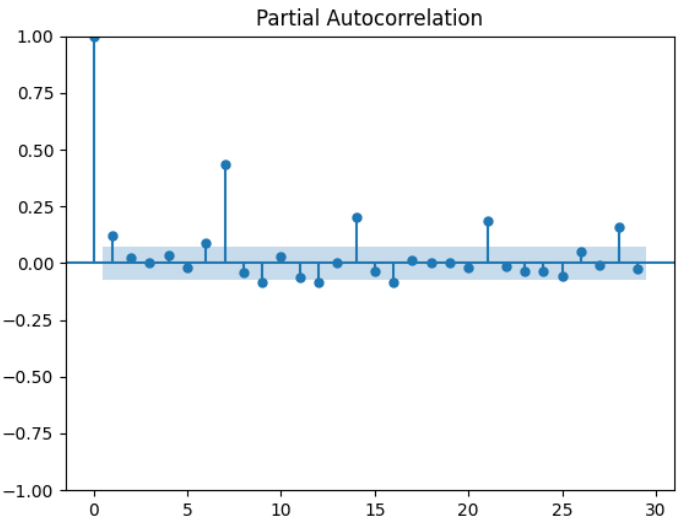
Función de Autocorrelación de la variable que representa el Total de ventas diarias de la Cevichería Rico Norte.



Nota: Elaboración propia

Figura 18

Función de Autocorrelación Parcial de la variable que representa el Total de ventas diarias de la Cevichería Rico Norte.



Nota: Elaboración propia

Se puede apreciar en las Figuras 17 y 18 que el primer rezago es muy alto, lo que sugiere un valor de 0 o 1 para $\boldsymbol{q}$ y para $\boldsymbol{p}$.

Teniendo listos los parámetros para el modelo, se procedió a separar el *dataframe* en entrenamiento y prueba, y a ajustar el modelo según los parámetros obtenidos.

```
train_size = int(len(X) * 0.8)  #80% para entrenamiento
train, test = X[0:train_size], X[train_size:len(X)]

p, d, q = 1, 0, 0
P, D, Q, s = 1, 0, 0, 7
```

Se consideró al valor s ($\boldsymbol{m}$) para que realice una estacionariedad semanal.

Se creó un *dataframe* con características exógenas para el conjunto de entrenamiento. Posteriormente, se entrenó el modelo SARIMA.

```
exogenous_train = train[['Dia_de_semana', 'Mes', 'Es_fin_de_semana',
'Lag_1', 'Media_movil_7']]

# Fit el modelo SARIMA con variables exógenas
model = SARIMAX(train[variable],
                order=(p, d, q),
                seasonal_order=(P, D, Q, s),
                exog=exogenous_train)
results = model.fit()
```

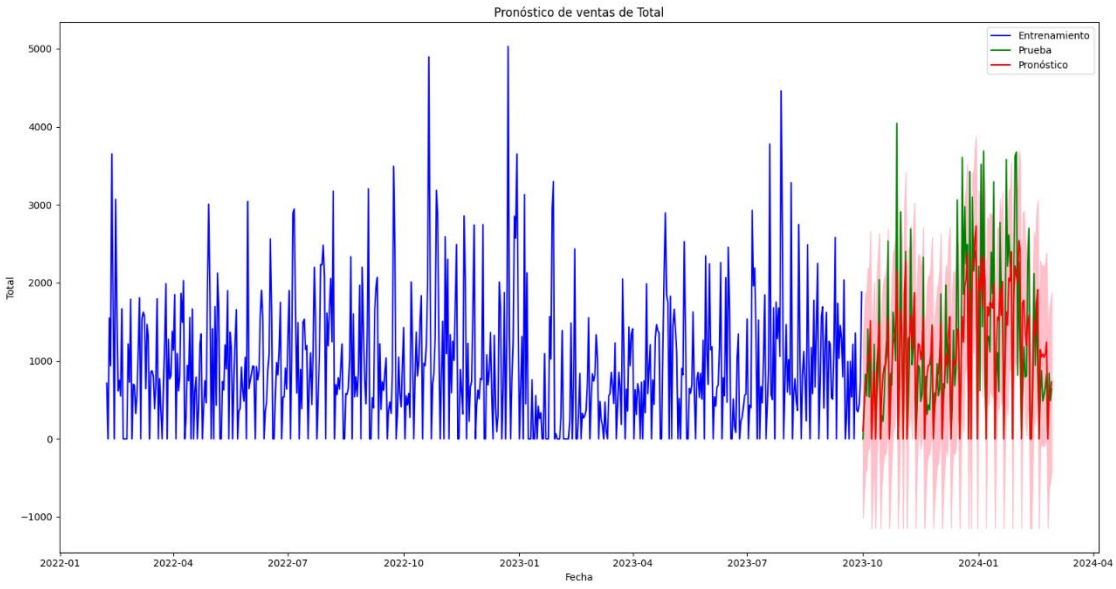
Para graficar el pronóstico, se usó el siguiente script.

```
plt.figure(figsize=(12, 6))
plt.plot(train[variable], label='Entrenamiento', color='blue')
plt.plot(test[variable], label='Prueba', color='green')
plt.plot(media_pronostico, label='Pronóstico', color='red')
plt.fill_between(pronostico_ci.index, pronostico_ci.iloc[:, 0],
pronostico_ci.iloc[:, 1], color='pink')
plt.title("Pronóstico de ventas de " + variable)
plt.xlabel("Fecha")
plt.ylabel(variable)
```

```
plt.legend()
plt.show()
```

Figura 19

Pronóstico de ventas con modelo SARIMA de Total.



Nota: Elaboración propia

Se observa en la Figura 19 que los datos de pronóstico no son muy similares a los datos de prueba. A continuación, se muestran las métricas de rendimiento.

```
print(df[[variable]].describe())

mae = mean_absolute_error(test[variable], media_pronostico)
mse = mean_squared_error(test[variable], media_pronostico)
r2 = r2_score(test[variable], media_pronostico)
def modified_mape(y_true, y_pred):
    mask = y_true != 0
    return np.mean(np.abs((y_true[mask] - y_pred[mask]) /
y_true[mask])) * 100

mape = modified_mape(test[variable], media_pronostico)

print(f'MAE: {mae}')
print(f'MSE: {mse}')
print(f'R²: {r2}')
print(f'MAPE: {mape:.2f}%')
```

Figura 20

Métricas de rendimiento del modelo SARIMA para la variable que representa el Total de ventas diarias de la Cevichería Rico Norte.

```
              Total
count    597.000000
mean    1235.516750
std      825.827029
min       67.000000
25%      625.000000
50%     1005.000000
75%     1656.000000
max     5029.500000
MAE: 436.64202801648327
MSE: 391553.38470515097
R²: 0.6209143304623442
MAPE: 48.49%
```

Nota: Elaboración propia

Según la Figura 20, las métricas de rendimiento no arrojan buenos resultados, sobre todo por el R^2 y el MAPE.

Se repitió el proceso, pero ahora eliminando los valores atípicos.

```
#Calcular Z-scores
z_scores = stats.zscore(df[variable])

#Eliminar outliers
df= df[(z_scores >= -3) & (z_scores <= 3)]
```

Figura 21

Métricas de rendimiento del modelo SARIMA para la variable que representa el Total de ventas diarias de la Cevichería Rico Norte con Outliers eliminados.

```
              Total
count    592.000000
mean    1208.439189
std      773.383732
min       67.000000
25%      624.000000
50%     1001.000000
75%     1622.875000
max     3687.500000
MAE: 421.94605799346533
MSE: 378247.93945082946
R²: 0.6168157268366812
MAPE: 47.09%
```

Nota: Elaboración propia

Como se puede apreciar en la Figura 21, las métricas de rendimiento casi no tuvieron variación al eliminar los valores atípicos. A menos que se hagan pruebas más exhaustivas, se puede ir concluyendo que SARIMA no es un buen modelo de pronóstico para los datos.

Se repitió el mismo proceso para *producto_16, producto_17*, etc., solo considerando las métricas de rendimiento.

XGBOOST:

Para realizar el modelado de XGBoost, se usaron de principal apoyo las librerías *xgboost* y *scikitlearn*.

Se volvió a usar la variable *Total* como prueba. A continuación, se dividen los datos de entrenamiento y prueba.

```
#Variables predictoras
X = df[['Dia_del_año', 'Año', 'Mes', 'Dia_de_semana',
'Es_fin_de_semana', 'Dia_del_mes', 'Semana_del_año', 'Trimestre',
'Lag_1', 'Lag_7', 'Media_movil_3', 'Media_movil_7', 'Media_movil_30',
'Diferencia_dia_anterior']]

#Variable objetivo
y = df[variable]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=

0.2, random_state=42)
```

Se usaron los siguientes parámetros para entrenar el modelo para la variable *Total*.

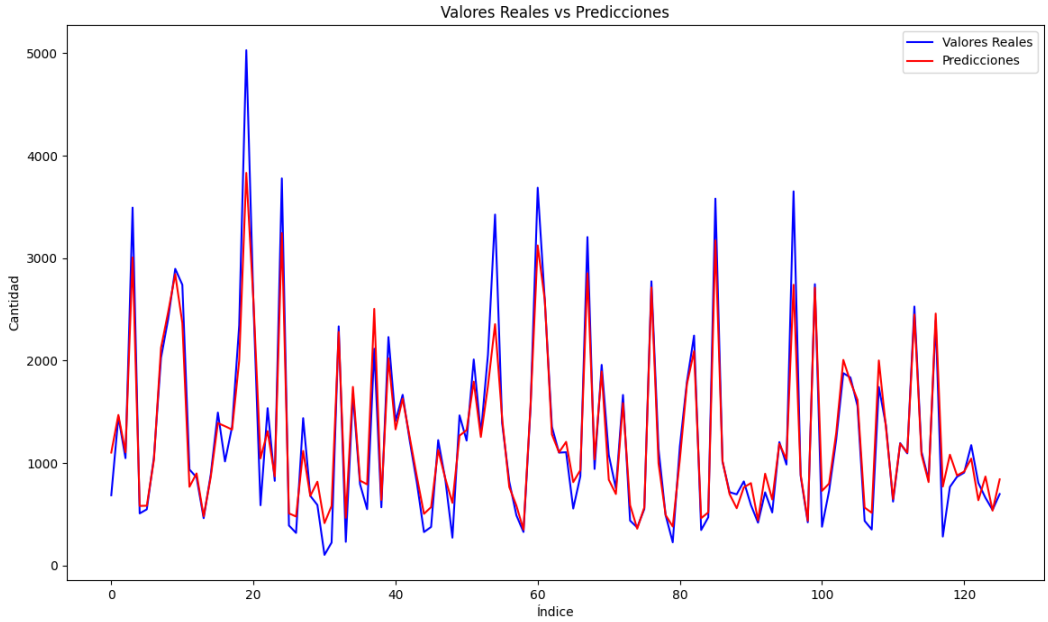
```
model= xgb.XGBRegressor(
        objective='reg:squarederror',
        alpha= 0,
        colsample_bytree= 1.0,
        subsample= 0.7,
        learning_rate= 0.01,
        n_estimators=250,
        seed=42,
        max_depth= 5)


model.fit(X_train, y_train)
```

Se Graficaron las predicciones del modelo.

```
plt.figure(figsize=(14, 8))
plt.plot(y_test.values, label='Valores Reales', color='b')
plt.plot(y_test_pred, label='Predicciones', color='r')
plt.xlabel('Índice')
plt.ylabel('Cantidad')
plt.title('Valores Reales vs Predicciones')
plt.legend()
plt.show()
```

Figura 22

Valores reales del Total de ventas diarias vs sus predicciones.



Nota: Elaboración propia

Se aprecia que el modelo aparenta capturar considerablemente bien el comportamiento de los datos reales.

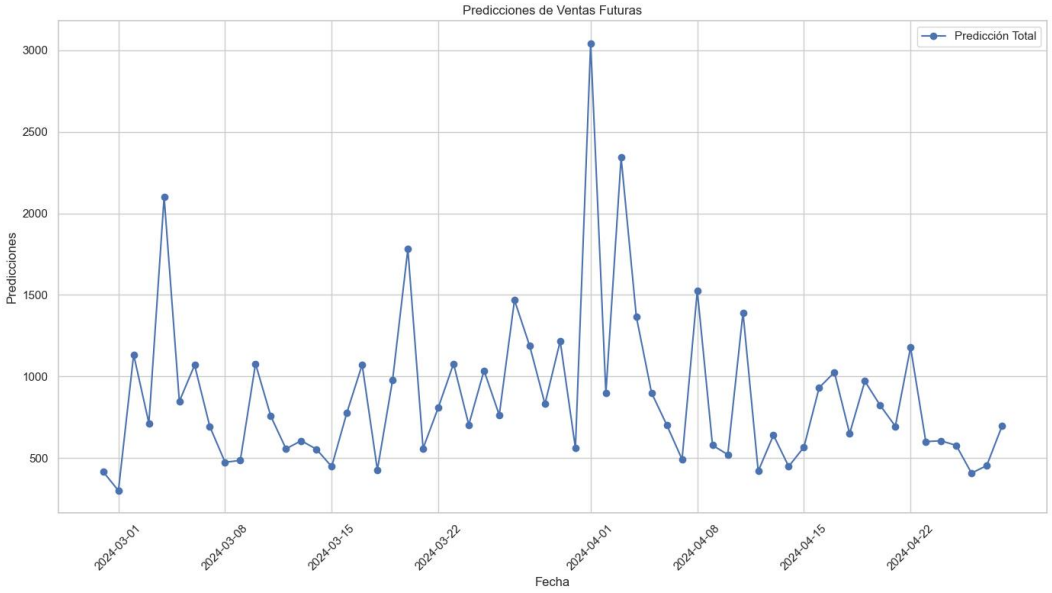
Se graficó un ejemplo de predicción para 2 meses.

```
dias_futuros = pd.date_range(start='2024-02-29', periods=60)
futuro_df = pd.DataFrame(dias_futuros, columns=['Fecha'])

futuro_df['Dia_del_año'] = futuro_df['Fecha'].dt.dayofyear
futuro_df['Año'] = futuro_df['Fecha'].dt.year
futuro_df['Mes'] = futuro_df['Fecha'].dt.month
futuro_df['Dia_de_semana'] = futuro_df['Fecha'].dt.dayofweek
futuro_df['Es_fin_de_semana'] = futuro_df['Dia_de_semana'].isin([5,
6]).astype(int)

futuro_df['Dia_del_mes'] = df['Fecha'].dt.day
futuro_df['Semana_del_año'] = df['Fecha'].dt.isocalendar().week
futuro_df['Trimestre'] = df['Fecha'].dt.quarter

futuro_df['Media_movil_3'] = df[variable].rolling(window=3).mean()
futuro_df['Media_movil_7'] = df[variable].rolling(window=7).mean()
futuro_df['Media_movil_30'] = df[variable].rolling(window=30).mean()
futuro_df['Diferencia_dia_anterior'] = df[variable].diff()
```

```
last_known_values = df[['Fecha',
variable]].sort_values(by='Fecha').tail(7)
futuro_df = futuro_df.merge(last_known_values, how='left', on='Fecha')

futuro_df['Lag_1'] = futuro_df[variable].shift(1)
futuro_df['Lag_7'] = futuro_df[variable].shift(7)
futuro_df.fillna(0, inplace=True)  #Rellenar NaNs con 0

X_futuro = futuro_df[['Dia_del_año', 'Año', 'Mes', 'Dia_de_semana',
'Es_fin_de_semana', 'Dia_del_mes', 'Semana_del_año', 'Trimestre',
'Lag_1', 'Lag_7', 'Media_movil_3', 'Media_movil_7', 'Media_movil_30',
'Diferencia_dia_anterior']]

pred_futuro = model.predict(X_futuro)

futuro_df[variable + '_Pred'] = pred_futuro

print(futuro_df)

sns.set(style="whitegrid")

#Crear el gráfico
plt.figure(figsize=(14, 8))

plt.plot(futuro_df['Fecha'], futuro_df[variable + '_Pred'],
marker='o', linestyle='-', color='b', label='Predicción '+ variable)

plt.xlabel('Fecha')
plt.ylabel('Predicciones')
plt.title('Predicciones de Ventas Futuras')
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Figura 23

Predicción de 2 meses del Total de ventas diarias de la Cevichería Rico Norte.



Nota: Elaboración propia

Para saber si realmente las predicciones son buenas, se evaluó el modelo con las métricas de rendimiento.

```
y_pred = model.predict(X_test)

r2= r2_score(y_test, y_pred)

mae = mean_absolute_error(y_test, y_pred)

mse= mean_squared_error(y_test, y_pred)

#MSE en el conjunto de entrenamiento
y_train_pred = model.predict(X_train)
mse_train = mean_squared_error(y_train, y_train_pred)

#MSE en el conjunto de prueba
y_test_pred = model.predict(X_test)
mse_test = mean_squared_error(y_test, y_test_pred)

#Validación cruzada
cv_scores = cross_val_score(model, X, y, cv=5,
scoring='neg_mean_squared_error')
cv_mse = -cv_scores.mean()
#MAPE
```

```
mape = mean_absolute_percentage_error(y_test, y_test_pred)

print(df[[variable]].describe())
print("R2 de " + variable +": ", r2)
print(f'MAE en el conjunto de prueba: {mae}')
print(f'MSE en el conjunto de entrenamiento: {mse_train}')
print(f'MSE en el conjunto de prueba: {mse_test}')
print(f'MSE con validación cruzada: {cv_mse}')
print(f'MAPE: {mape}')
```

Figura 24

Métricas de rendimiento del modelo XGBoost para el Total de ventas diarias.

```
              Total
count    626.000000
mean    1241.110224
std      817.338942
min       67.000000
25%      636.125000
50%     1016.750000
75%     1658.250000
max     5029.500000
R2 de  Total :  0.9289551450860961
MAE en el conjunto de prueba: 154.0688263423859
MSE en el conjunto de entrenamiento: 11387.356668243574
MSE en el conjunto de prueba: 61186.54938266336
MSE con validación cruzada: 50647.277440979415
MAPE: 0.202491489368792
```

Nota: Elaboración propia

Como se puede observar en la Figura 24, el R^2 es alto e indica que puede explicar alrededor del 93% de la variabilidad de, en este caso, el Total diario.

El MAE indica que las predicciones del *Total* tienen un error absoluto de aproximadamente 154 soles.

El MSE del conjunto de prueba es más alto que el de entrenamiento, esto puede indicar sobre entrenamiento (*overfitting*).

El MAPE indica que las predicciones se desvían alrededor del 20% respecto a los valores reales.

Se evaluó si estos valores se pueden mejorar, para ello se repitió el mismo proceso, pero esta vez eliminando los valores atípicos.

```
#Calcular Z-scores
z_scores = stats.zscore(df['Total'])

#Eliminar outliers
df= df[(z_scores >= -3) & (z_scores <= 3)]
```

Figura 25

Métricas de rendimiento del modelo XGBoost para el Total de ventas diarias con outliers eliminados.

```
              Total
count    621.000000
mean    1215.342190
std      767.013323
min       67.000000
25%      635.500000
50%     1005.500000
75%     1624.000000
max     3687.500000
R2 de  Total :  0.9293074816460148
MAE en el conjunto de prueba: 133.45464599609375
MSE en el conjunto de entrenamiento: 9385.285580640619
MSE en el conjunto de prueba: 39788.590121480316
MSE con validación cruzada: 44866.23238213808
MAPE: 0.15947313514796388
```

Nota: Elaboración propia

Como se puede apreciar en la Figura 25, el R^2 se mantiene en un nivel alto.

El MAE disminuyó a 133.45 soles, indicando que el modelo es más preciso.

El MSE en el conjunto de prueba disminuyó considerablemente.

El MAPE ahora indica que las predicciones se desvían alrededor del 16% respecto a los valores reales.

El modelo mejoró notablemente al eliminar los valores atípicos, y podemos apreciar que XGBoost tuvo un desempeño mucho mayor que SARIMA.

Se repitió el mismo proceso para *producto_16, producto_17*, etc., solo considerando las métricas de rendimiento.

LSTM:

Para el modelado de una Red Neuronal Recurrente LSTM, se necesitó principalmente de las librerías *scikitlearn* y *tensorflow*.

Nuevamente se volvió a usar la variable *Total* como prueba.

Para poder modelar el LSTM, fue necesario escalar los datos y crear un nuevo conjunto de datos.

```
#Escalar
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_series = scaler.fit_transform(series.values.reshape(-1, 1))

#Secuencia de datos para LSTM
def create_dataset(data, time_step=1):
    X, Y = [], []
    for i in range(len(data) - time_step - 1):
        a = data[i:(i + time_step), 0]
        X.append(a)
        Y.append(data[i + time_step, 0])
    return np.array(X), np.array(Y)

time_step = 10
X, Y = create_dataset(scaled_series, time_step)

#División de los datos de entrenamiento y prueba
```

```
train_size = int(len(X) * 0.8)
test_size = len(X) - train_size
X_train, X_test = X[0:train_size], X[train_size:len(X)]
Y_train, Y_test = Y[0:train_size], Y[train_size:len(Y)]

X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)

#Construcción del modelo

model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(time_step, 1)))
model.add(LSTM(50, return_sequences=False))
model.add(Dense(1))


model.compile(optimizer='adam', loss='mean_absolute_error')

early_stopping = EarlyStopping(monitor='val_loss', patience=10)
model.fit(X_train, Y_train, epochs=100, batch_size=32,
validation_data=(X_test, Y_test), callbacks=[early_stopping],
verbose=1)
```

Se empezó usando el optimizador “Adam” y la función de pérdida MAE.

El entrenamiento empezó con 100 épocas y un lote de 32.

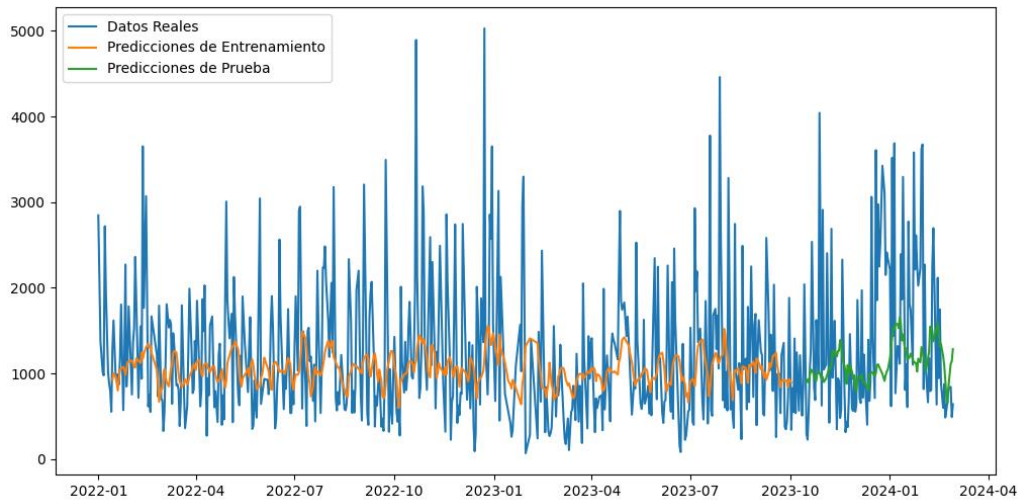
Se graficaron las predicciones.

```
train_predict = model.predict(X_train)
test_predict = model.predict(X_test)

train_predict = scaler.inverse_transform(train_predict)
test_predict = scaler.inverse_transform(test_predict)

Y_train = scaler.inverse_transform([Y_train])
Y_test = scaler.inverse_transform([Y_test])


plt.figure(figsize=(12, 6))
plt.plot(series.index, series, label='Datos Reales')
train_predict_plot = np.empty_like(series)
train_predict_plot[:] = np.nan
train_predict_plot[time_step:len(train_predict) + time_step] =
train_predict[:, 0]

test_predict_plot = np.empty_like(series)
```

```
test_predict_plot[:] = np.nan

test_start = len(train_predict) + (time_step * 2)
test_end = test_start + len(test_predict)

if test_end > len(test_predict_plot):
    test_end = len(test_predict_plot)

test_predict_plot[test_start:test_end] = test_predict[:test_end -
test_start, 0]

plt.plot(series.index, train_predict_plot, label='Predicciones de
Entrenamiento')
plt.plot(series.index, test_predict_plot, label='Predicciones de
Prueba')
plt.legend()
plt.show()
```

Figura 26

Valores predichos con LSTM para el Total de ventas diarias.



Nota: Elaboración propia

Como se aprecia en la Figura 26, las predicciones aparentan estar muy alejadas de los patrones de los datos reales.

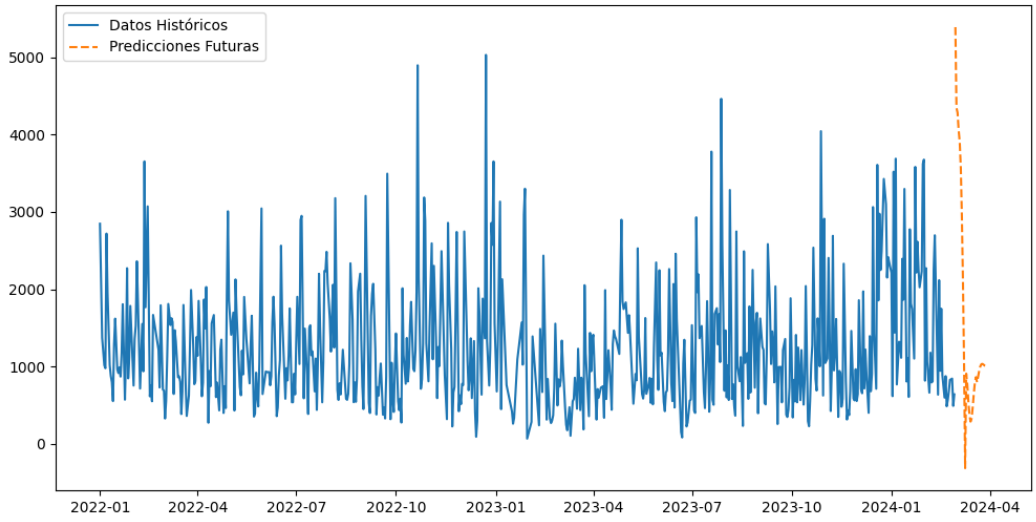
Se graficó una predicción para 30 días.

```
x_input = test_predict[-time_step:].reshape(1, -1)
temp_input = list(x_input)
temp_input = temp_input[0].tolist()
```

```
lst_output = []
n_steps = time_step
i = 0
while(i < 30):
    if(len(temp_input) > time_step):
        x_input = np.array(temp_input[1:])
        x_input = x_input.reshape((1, time_step, 1))
        yhat = model.predict(x_input, verbose=0)
        temp_input.extend(yhat[0].tolist())
        temp_input = temp_input[1:]
        lst_output.extend(yhat.tolist())
        i = i + 1
    else:
        x_input = x_input.reshape((1, time_step, 1))
        yhat = model.predict(x_input, verbose=0)
        temp_input.extend(yhat[0].tolist())
        lst_output.extend(yhat.tolist())
        i = i + 1

future_predictions = scaler.inverse_transform(lst_output)

future_dates = pd.date_range(start=series.index[-1] +
pd.Timedelta(days=1), periods=30, freq='D')
future_df = pd.DataFrame(future_predictions, index=future_dates,
columns=['Predicción'])

plt.figure(figsize=(12, 6))
plt.plot(series.index, series, label='Datos Históricos')
plt.plot(future_df, label='Predicciones Futuras', linestyle='dashed')
plt.legend()
plt.show()
```

Figura 27

Predicciones del Total de ventas diarias con LSTM para 30 días.



Nota: Elaboración propia

Como se puede apreciar en la Figura 27, las predicciones están muy fuera de lugar. A continuación, se observan las métricas de rendimiento.

```
#MSE
train_mse = np.mean((Y_train[0] - train_predict[:, 0])**2)
test_mse = np.mean((Y_test[0] - test_predict[:, 0])**2)

#R² para el conjunto de entrenamiento y prueba
train_r2 = r2_score(Y_train[0], train_predict[:, 0])
test_r2 = r2_score(Y_test[0], test_predict[:, 0])

#MAE para el conjunto de entrenamiento y prueba
train_mae = mean_absolute_error(Y_train[0], train_predict[:, 0])
test_mae = mean_absolute_error(Y_test[0], test_predict[:, 0])

#MAPE para el conjunto de entrenamiento y prueba
def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

train_mape = mean_absolute_percentage_error(Y_train[0],
train_predict[:, 0])
test_mape = mean_absolute_percentage_error(Y_test[0], test_predict[:,
0])

print(f'R² en el conjunto de entrenamiento: {train_r2}')
```

```
print(f'R² en el conjunto de prueba: {test_r2}')
print(f'MAE en el conjunto de entrenamiento: {train_mae}')
print(f'MAE en el conjunto de prueba: {test_mae}')
print(f'MSE en el conjunto de entrenamiento: {train_mse}')
print(f'MSE en el conjunto de prueba: {test_mse}')
print(f'MAPE en el conjunto de entrenamiento: {train_mape}')
print(f'MAPE en el conjunto de prueba: {test_mape}')
```

Figura 28

Métricas de rendimiento del modelo LSTM para el Total de ventas diarias.

```
R² en el conjunto de entrenamiento: -0.025414711951796898
R² en el conjunto de prueba: -0.003281758081161712
MAE en el conjunto de entrenamiento: 560.1097993928242
MAE en el conjunto de prueba: 675.9767277415206
MSE en el conjunto de entrenamiento: 613319.4422605794
MSE en el conjunto de prueba: 900494.0159548948
MAPE en el conjunto de entrenamiento: 68.84838122007159
MAPE en el conjunto de prueba: 51.137712889045076
```

Nota: Elaboración propia

La Figura 28 muestra valores extremadamente malos de R^2.

MAE y MSE son muy altos, indicando que las predicciones se alejan mucho de los valores reales.

El MAPE indica que las predicciones se desvían mucho de los valores reales.

Se repitió el mismo proceso, pero eliminando los valores atípicos.

Figura 29

Métricas de rendimiento del modelo LSTM para el Total de ventas diarias con outliers eliminados.

```
R² en el conjunto de entrenamiento: -0.06642924188850885
R² en el conjunto de prueba: -0.04537615735817946
MAE en el conjunto de entrenamiento: 547.3362166607966
MAE en el conjunto de prueba: 672.9259813652664
MSE en el conjunto de entrenamiento: 541968.1160908401
MSE en el conjunto de prueba: 887470.100860796
MAPE en el conjunto de entrenamiento: 66.08329303967875
MAPE en el conjunto de prueba: 49.707762992815255
```

Nota: Elaboración propia

Como se ve en la Figura 29, la eliminación de los valores atípicos casi no afectó a los resultados.

Se probaron otros optimizadores, como *RMSprop* y *SGD*, así como números diferentes de épocas y tamaño de lote, pero los resultados no diferían significativamente o incluso empeoraban.

Hasta el momento, *XGBoost* sigue siendo considerado el mejor modelo para estos datos.

Se repitió el mismo proceso para *producto_16, producto_17*, etc., solo considerando las métricas de rendimiento.

MLP:

Para el Modelo de Perceptrón Multicapa, se utilizó en su mayoría la librería scikitlearn.

Se volvió a trabajar con la variable que representa el Total de ventas diarias.

Al igual que LSTM, fue necesario escalar los datos y crear un nuevo conjunto de datos.

```
#Escalar
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_series = scaler.fit_transform(series.values.reshape(-1, 1))

#Crear secuencia de datos para MLP
def create_dataset(data, time_step=1):
    X, Y = [], []
    for i in range(len(data) - time_step - 1):
        a = data[i:(i + time_step), 0]
        X.append(a)
        Y.append(data[i + time_step, 0])
    return np.array(X), np.array(Y)

time_step = 10
X, Y = create_dataset(scaled_series, time_step)

#División de datos en entrenamiento y prueba
train_size = int(len(X) * 0.8)
test_size = len(X) - train_size
X_train, X_test = X[0:train_size], X[train_size:len(X)]
Y_train, Y_test = Y[0:train_size], Y[train_size:len(Y)]

#Creación y entrenamiento del modelo MLP
mlp = MLPRegressor(hidden_layer_sizes=(100, 100), alpha=0.0001,
learning_rate_init=0.001, max_iter=2000, random_state=42)

mlp.fit(X_train, Y_train)
```

Se definió la tasa de aprendizaje inicialmente a 0.001, con un máximo de iteraciones de 2000, una regularización L2 (Alpha), y 2 capas con 100 neuronas cada una.

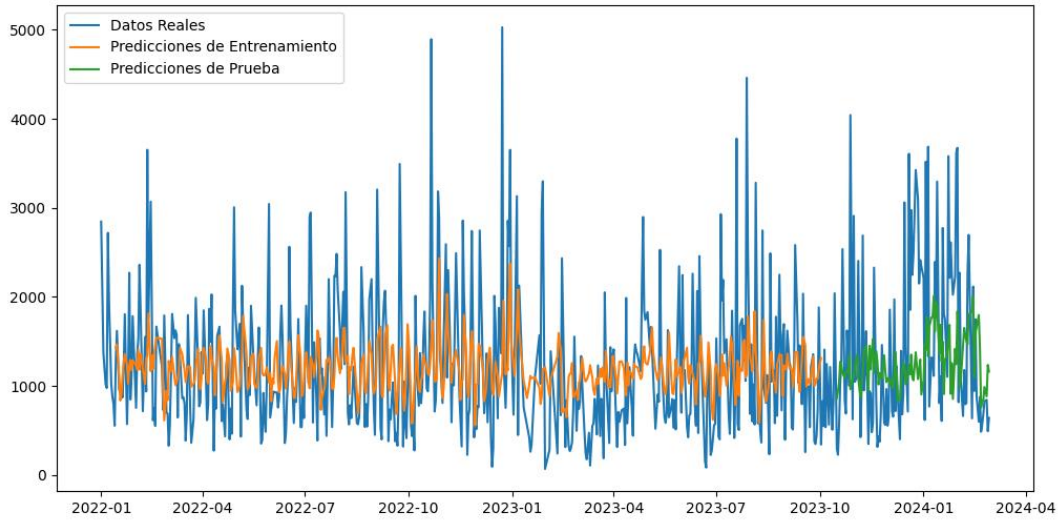
Se procedió a visualizar las predicciones.

```
train_predict = mlp.predict(X_train)
test_predict = mlp.predict(X_test)
train_predict = scaler.inverse_transform(train_predict.reshape(-1, 1))
test_predict = scaler.inverse_transform(test_predict.reshape(-1, 1))

Y_train = scaler.inverse_transform(Y_train.reshape(-1, 1))
Y_test = scaler.inverse_transform(Y_test.reshape(-1, 1))
```

```
plt.figure(figsize=(12, 6))
plt.plot(series.index, series, label='Datos Reales')

train_predict_plot = np.empty_like(series, dtype=float)
train_predict_plot[:] = np.nan
train_predict_plot[time_step:len(train_predict) + time_step] =
train_predict[:, 0]
test_predict_plot = np.empty_like(series, dtype=float)
test_predict_plot[:] = np.nan

test_start = len(train_predict) + (time_step * 2)
test_end = test_start + len(test_predict)

if test_end > len(test_predict_plot):
    test_end = len(test_predict_plot)

test_predict_plot[test_start:test_end] = test_predict[:test_end -
test_start, 0]

plt.plot(series.index, train_predict_plot, label='Predicciones de
Entrenamiento')
plt.plot(series.index, test_predict_plot, label='Predicciones de
Prueba')
plt.legend()
plt.show()
```

Figura 30

Valores predichos con MLP para el Total de ventas diarias.



Nota: Elaboración propia

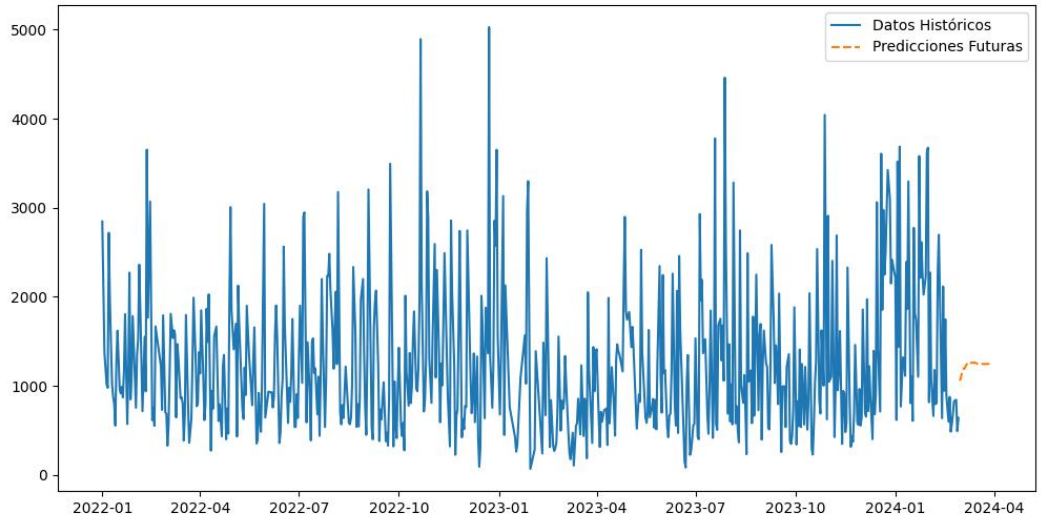
En la Figura 30 se puede ver que, similar a LSTM, las predicciones aparentan estar muy alejadas de los patrones de los datos reales.

Se graficaron las predicciones para 30 días.

```
X_input = X_test[-1]
temp_input = list(x_input)

lst_output = []
i = 0
while(i < 30):
    if(len(temp_input) > time_step):
        x_input = np.array(temp_input[1:])
        x_input = x_input.reshape((1, time_step))
        yhat = mlp.predict(x_input)
        temp_input.extend(yhat.tolist())
        temp_input = temp_input[1:]
        lst_output.extend(yhat.tolist())
        i = i + 1
    else:
        x_input = x_input.reshape((1, time_step))
        yhat = mlp.predict(x_input)
        temp_input.extend(yhat.tolist())
        lst_output.extend(yhat.tolist())
        i = i + 1

future_predictions =
scaler.inverse_transform(np.array(lst_output).reshape(-1, 1))

future_dates = pd.date_range(start=series.index[-1] +
pd.Timedelta(days=1), periods=30, freq='D')
future_df = pd.DataFrame(future_predictions, index=future_dates,
columns=['Predicción'])

plt.figure(figsize=(12, 6))
plt.plot(series.index, series, label='Datos Históricos')
plt.plot(future_df, label='Predicciones Futuras', linestyle='dashed')
plt.legend()
plt.show()
```

Figura 31

Predicciones del Total de ventas diarias con MLP para 30 días.



Nota: Elaboración propia

Como se puede apreciar en la Figura 31, las predicciones no son para nada significativas. A continuación, se aprecian las métricas de rendimiento.

```
#Conjunto de entrenamiento
train_mse = mean_squared_error(Y_train, train_predict)
train_mae = mean_absolute_error(Y_train, train_predict)
train_mape = mean_absolute_percentage_error(Y_train, train_predict)
r2_train = r2_score(Y_train, train_predict)

#Conjunto de prueba
test_mse = mean_squared_error(Y_test, test_predict)
test_mae = mean_absolute_error(Y_test, test_predict)
test_mape = mean_absolute_percentage_error(Y_test, test_predict)
r2_test = r2_score(Y_test, test_predict)

print(f'MSE en el conjunto de entrenamiento: {train_mse}')
print(f'MAE en el conjunto de entrenamiento: {train_mae}')
print(f'MAPE en el conjunto de entrenamiento: {train_mape}')
print(f'R2 en el conjunto de entrenamiento: {r2_train}')

print(f'\nMSE en el conjunto de prueba: {test_mse}')
print(f'MAE en el conjunto de prueba: {test_mae}')
print(f'MAPE en el conjunto de prueba: {test_mape}')
print(f'R2 en el conjunto de prueba: {r2_test}')
```

Figura 32

Métricas de rendimiento del modelo MLP para el Total de ventas diarias.

```
MSE en el conjunto de entrenamiento: 519757.33105483407
MAE en el conjunto de entrenamiento: 531.975693884287
MAPE en el conjunto de entrenamiento: 0.7467934726565753
R2 en el conjunto de entrenamiento: 0.13101268737868066

MSE en el conjunto de prueba: 847751.4093309357
MAE en el conjunto de prueba: 698.2563212664182
MAPE en el conjunto de prueba: 0.6255923512231697
R2 en el conjunto de prueba: 0.05548120331770601
```

Nota: Elaboración propia

Como se ve en la Figura 32, R^2 es demasiado bajo, MAE y MSE indican que existe un sobreajuste en el modelo, y el MAPE indica que las predicciones se desvían mucho de los valores reales, similar al modelo LSTM.

Se realizó el mismo procedimiento, pero esta vez eliminando los valores atípicos.

Figura 33

Métricas de rendimiento del modelo MLP para el Total de ventas diarias con outliers eliminados.

```
MSE en el conjunto de entrenamiento: 430572.8447096135
MAE en el conjunto de entrenamiento: 502.83689725668034
MAPE en el conjunto de entrenamiento: 0.7333107864538316
R2 en el conjunto de entrenamiento: 0.15276294171428972

MSE en el conjunto de prueba: 788971.0935788866
MAE en el conjunto de prueba: 675.8067329315247
MAPE en el conjunto de prueba: 0.6198259113959925
R2 en el conjunto de prueba: 0.07064861196766514
```

Nota: Elaboración propia

Como se aprecia en la Figura 33, casi no hay diferencia al eliminar los valores atípicos.

Se probó aumentar las capas y neuronas, así como el número de iteraciones, pero los resultados siguieron siendo malos.

A menos que se realicen pruebas exhaustivas, el modelo MLP no es adecuado para los datos.

4.1.5. Evaluación

En esta fase se evaluaron y compararon los resultados de las métricas de rendimiento de los modelos vistos anteriormente.

Todos los modelos se realizaron eliminando los valores atípicos.

Tabla 4

Métricas de rendimiento de SARIMA para el Total de ventas diarias y los 25 productos más vendidos diariamente de la Cevichería Rico Norte.

	SARIMA			
	MAE	**MSE**	**R2**	**MAPE**
Total	421.946058	378247.939	0.61681573	47.09%
producto_16 (Combinado de 14)	6.09231192	78.3197575	0.43813005	118.39%
producto_17 (Combinado de 18)	2.79118209	18.2910197	0.5135619	104.08%
producto_26 (Dúo Clásico)	2.92521307	21.7035653	0.34378418	63.49%
producto_21 (Triple Tradicional)	3.46695569	33.0090423	0.18677444	164.88%
producto_49 (Gaseosa 1 Litro)	1.42157893	4.18504819	0.55613774	68.63%
producto_54 (Gaseosa Personal)	1.56828184	4.60562528	0.40646556	79.27%
producto_2 (Ceviche de Pescado)	1.14629492	2.65903115	0.48027901	68.93%

	SARIMA			
	MAE	**MSE**	**R2**	**MAPE**
producto_45 (Leche de Tigre de 5)	1.23890373	2.8201889	0.39858922	66.81%
producto_57 (Cerveza Cuzqueña)	0.94549517	1.69368689	0.47467414	54.25%
producto_48 (Jarra de Chicha)	1.04522337	2.31827561	0.31747279	57.26%
producto_53 (Gaseosa Gordita)	1.03080314	1.98065543	0.38623659	62.11%
producto_31 (Chicharrón de Pota-Porción)	0.95567912	1.80951156	0.53250081	60.06%
producto_52 (Gaseosa medio Litro)	0.92991245	2.02628571	0.35533394	57.38%
producto_51 (Gaseosa 1 Litro y medio)	0.75443502	1.21126267	0.5422785	49.28%
producto_55 (Chicha 2 Litros)	0.55639323	0.75751315	0.48994356	38.89%
producto_7 (Ceviche Clásico + Chicharrón de Pescado, Pota o Mixto)	0.60309805	0.72427474	0.46742169	43.21%
producto_46 (Leche de Tigre de 12)	0.53802042	0.59569874	0.52301702	34.30%
producto_37 (Chaufa Tradicional + Chicharrón de Pescado o Mixto)	0.50221556	0.53856335	0.38253884	34.76%
producto_27 (Ronda Marina (1))	0.509658	0.5062991	0.42000256	36.60%
producto_28 (Ronda Marina (2))	0.49776232	0.54395296	0.39973519	36.30%
producto_56 (Chicha Personal)	0.60216065	0.76054674	0.38567287	40.16%
producto_22 (Triple a la Crema o Macho)	0.47259807	0.46610381	0.51411529	34.69%
producto_38 (Chaufa Tradicional + Chicharrón de Pota)	0.48053583	0.45087256	0.35376256	35.23%
producto_5 (Ceviche de Pescado + Chicharrón de Pescado, Pota o Mixto)	0.41834038	0.33151099	0.27448717	34.83%

	SARIMA			
	MAE	**MSE**	**R2**	**MAPE**
producto_18 (Combinado con Mariscos)	0.60875391	1.13555698	0.35710794	46.07%

Nota: Elaboración propia

Como se puede apreciar en la Tabla 4, el modelo SARIMA obtuvo un MAE alto en la variable Total, indicando que la predicción de las ventas diarias se desvía en 421.94 soles de los valores reales. También se aprecia un MSE alto con respecto a la desviación estándar de las variables, con un $\boldsymbol{R^2}$ considerablemente bajo, así como un alto porcentaje del MAPE, lo que puede ocasionar predicciones erróneas.

Tabla 5

Métricas de rendimiento de XGBOOST para el Total de ventas diarias y los 25 productos más vendidos diariamente de la Cevichería Rico Norte.

	XGBOOST			
	MAE	**MSE**	**R2**	**MAPE**
Total	133.454646	39788.5901	0.92930748	15.95%
producto_16 (Combinado de 14)	1.88803145	11.1353137	0.92335244	47.67%
producto_17 (Combinado de 18)	0.83937493	2.13797169	0.92915394	26.94%
producto_26 (Dúo Clásico)	0.55165487	1.67061817	0.94581043	118.29%
producto_21 (Triple Tradicional)	0.52328799	1.10655131	0.96579809	88.49%
producto_49 (Gaseosa 1 Litro)	0.35379324	0.28540335	0.95497269	12.61%
producto_54 (Gaseosa Personal)	0.38161092	0.41156971	0.95250757	51.36%
producto_2 (Ceviche de Pescado)	0.31081446	0.24791966	0.94497275	21.38%

	XGBOOST			
	MAE	**MSE**	**R2**	**MAPE**
producto_45 (Leche de Tigre de 5)	0.22936694	0.12317258	0.96869946	18.60%
producto_57 (Cerveza Cuzqueña)	0.19695324	0.10866238	0.96167048	22.13%
producto_48 (Jarra de Chicha)	0.24107298	0.23903782	0.914522	33.24%
producto_53 (Gaseosa Gordita)	0.18924336	0.14546056	0.9465835	29.11%
producto_31 (Chicharrón de Pota-Porción)	0.20760986	0.13354092	0.95056874	34.35%
producto_52 (Gaseosa medio Litro)	0.24463692	0.22815076	0.91697129	37.89%
producto_51 (Gaseosa 1 Litro y medio)	0.16825875	0.10598225	0.94987478	29.93%
producto_55 (Chicha 2 Litros)	0.19176484	0.1547673	0.92947366	34.20%
producto_7 (Ceviche Clásico + Chicharrón de Pescado, Pota o Mixto)	0.11635176	0.04439699	0.96944826	41.03%
producto_46 (Leche de Tigre de 12)	0.12828267	0.05322898	0.9542918	52.18%
producto_37 (Chaufa Tradicional + Chicharrón de Pescado o Mixto)	0.10368877	0.02771706	0.96370053	42.41%
producto_27 (Ronda Marina (1))	0.06851836	0.01907277	0.9627224	31.39%
producto_28 (Ronda Marina (2))	0.11841628	0.04018239	0.9653765	60.46%
producto_56 (Chicha Personal)	0.09558115	0.04855103	0.95646179	47.05%

	XGBOOST			
	MAE	**MSE**	**R2**	**MAPE**
producto_22 (Triple a la Crema o Macho)	0.0701246	0.03050708	0.95465302	33.88%
producto_38 (Chaufa Tradicional + Chicharrón de Pota)	0.08237617	0.02776757	0.94780512	49.76%
producto_5 (Ceviche de Pescado + Chicharrón de Pescado, Pota o Mixto)	0.06894472	0.01714501	0.969225	50.46%
producto_18 (Combinado con Mariscos)	0.13407602	0.19400822	0.8909839	143.98%

Nota: Elaboración propia

Como se ve en la Tabla 5, el modelo XGBoost obtuvo buenos resultados, con valores MAE y MSE bajos en todas las variables, y un $\boldsymbol{R^2}$ que explica un alto porcentaje de variabilidad de las variables. El MAPE obtenido es relativamente alto, solo unas pocas variables obtuvieron un MAPE por debajo del 20%.

Tabla 6

Métricas de rendimiento de LSTM para el Total de ventas diarias y los 25 productos más vendidos diariamente de la Cevichería Rico Norte.

	LSTM			
	MAE	**MSE**	**R2**	**MAPE**
Total	671.573055	868873.697	-0.02347093	48.26%
producto_16 (Combinado de 14)	8.71088699	171.578298	-0.1498902	89.16%

	LSTM			
	MAE	**MSE**	**R2**	**MAPE**
producto_17 (Combinado de 18)	4.18637529	39.8736265	-0.01093275	68.20%
producto_26 (Dúo Clásico)	4.00255181	44.6864927	-0.19493473	165.72%
producto_21 (Triple Tradicional)	3.84066426	55.7135882	-0.18935872	118.71%
producto_49 (Gaseosa 1 Litro)	2.15524383	7.56658755	0.04491219	55.21%
producto_54 (Gaseosa Personal)	2.11445464	8.88835671	-0.1261785	82.71%
producto_2 (Ceviche de Pescado)	1.71732966	5.26477257	-0.18153941	59.97%
producto_45 (Leche de Tigre de 5)	1.66749602	4.68352028	-0.15858124	61.32%
producto_57 (Cerveza Cuzqueña)	1.47193576	3.88795043	-0.34666388	71.29%
producto_48 (Jarra de Chicha)	1.43828061	4.36223557	-0.30214261	80.43%
producto_53 (Gaseosa Gordita)	1.39949846	4.3972129	-0.42622052	68.43%
producto_31 (Chicharrón de Pota-Porción)	1.58394667	5.44820466	-0.4715017	72.44%

	LSTM			
	MAE	**MSE**	**R2**	**MAPE**
producto_52 (Gaseosa medio Litro)	1.19026824	4.0290053	-0.26427765	71.15%
producto_51 (Gaseosa 1 Litro y medio)	1.26501007	3.64191324	-0.50769177	64.92%
producto_55 (Chicha 2 Litros)	0.67680987	1.59979474	-0.11250094	48.32%
producto_7 (Ceviche Clásico + Chicharrón de Pescado, Pota o Mixto)	0.78143373	1.46026282	-0.12983063	66.42%
producto_46 (Leche de Tigre de 12)	0.81360564	1.35196295	-0.19554572	70.37%
producto_37 (Chaufa Tradicional + Chicharrón de Pescado o Mixto)	0.57522714	0.85403005	-0.04775046	59.79%
producto_27 (Ronda Marina (1))	0.6194567	0.88039157	-0.08646435	62.69%
producto_28 (Ronda Marina (2))	0.60554365	0.87998557	-0.04559883	62.52%
producto_56 (Chicha Personal)	0.71612622	1.28294815	-0.03638536	73.27%
producto_22 (Triple a la Crema o Macho)	0.60990552	0.95610143	-0.0913988	57.71%
producto_38 (Chaufa Tradicional + Chicharrón de Pota)	0.53171524	0.65671437	-0.0080999	64.41%

	LSTM			
	MAE	**MSE**	**R2**	**MAPE**
producto_5 (Ceviche de Pescado + Chicharrón de Pescado, Pota o Mixto)	0.53550881	0.5514406	-0.17397171	89.86%
producto_18 (Combinado con Mariscos)	0.75938127	2.72439499	-0.26720139	199.85%

Nota: Elaboración propia

Como se aprecian en las métricas de la Tabla 6, el modelo LSTM obtuvo valores de MAE y MSE muy altos, R^2 negativos y MAPE con porcentajes muy altos. Este modelo está lejos de ser bueno, esto puede deberse a la naturaleza de los datos, o la falta de una configuración de parámetros avanzada del modelo.

Tabla 7

Métricas de rendimiento de MLP para el Total de ventas diarias y los 25 productos más vendidos diariamente de la Cevichería Rico Norte.

	MLP			
	MAE	**MSE**	**R2**	**MAPE**
Total	675.806733	788971.094	0.07064861	39.16%
producto_16 (Combinado de 14)	8.87502575	143.865224	0.03583837	85.22%
producto_17 (Combinado de 18)	4.58135958	32.916443	0.16545563	75.39%
producto_26 (Dúo Clásico)	4.83668668	46.6019193	-0.246154	145.14%
producto_21 (Triple Tradicional)	4.5488492	51.7551345	-0.10485471	124.57%

	MLP			
	MAE	**MSE**	**R2**	**MAPE**
producto_49 (Gaseosa 1 Litro)	2.45687301	9.50824102	-0.20017181	65.01%
producto_54 (Gaseosa Personal)	2.2875414	8.25439366	-0.04585369	84.89%
producto_2 (Ceviche de Pescado)	1.70174338	4.82600264	-0.08306906	58.69%
producto_45 (Leche de Tigre de 5)	1.98418889	6.44789936	-0.59504278	78.49%
producto_57 (Cerveza Cuzqueña)	1.89274112	5.60801541	-0.94244035	85.78%
producto_48 (Jarra de Chicha)	1.40275188	3.3246938	0.00756724	72.78%
producto_53 (Gaseosa Gordita)	1.43775609	3.46710215	-0.12454237	68.77%
producto_31 (Chicharrón de Pota-Porción)	1.5295178	3.72793435	-0.00687512	69.76%
producto_52 (Gaseosa medio Litro)	1.2933824	3.28524873	-0.03089131	75.13%
producto_51 (Gaseosa 1 Litro y medio)	1.62632912	4.41437601	-0.82747857	88.60%
producto_55 (Chicha 2 Litros)	0.85057308	1.43203229	0.00416145	57.97%

	MLP			
	MAE	**MSE**	**R2**	**MAPE**
producto_7 (Ceviche Clásico + Chicharrón de Pescado, Pota o Mixto)	0.89376721	1.32371789	-0.02418345	72.94%
producto_46 (Leche de Tigre de 12)	1.14240125	2.11298351	-0.86851895	94.72%
producto_37 (Chaufa Tradicional + Chicharrón de Pescado o Mixto)	0.8578435	1.36791971	-0.67820617	84.46%
producto_27 (Ronda Marina (1))	0.66975466	0.79434579	0.01972212	64.39%
producto_28 (Ronda Marina (2))	0.93806978	1.37053546	-0.62847019	88.08%
producto_56 (Chicha Personal)	0.79475549	1.25098542	-0.01056537	77.86%
producto_22 (Triple a la Crema o Macho)	0.88883679	1.34792744	-0.53867189	89.11%
producto_38 (Chaufa Tradicional + Chicharrón de Pota)	0.71975234	0.84616235	-0.29891506	92.76%
producto_5 (Ceviche de Pescado + Chicharrón de Pescado, Pota o Mixto)	0.77909411	0.92108232	-0.96090855	120.25%
producto_18 (Combinado con Mariscos)	0.81786231	2.13472377	0.00707315	161.83%

Nota: Elaboración propia

Como se aprecia en la Tabla 7, el modelo MLP obtuvo pésimos valores de MAE, MSE, $\boldsymbol{R^2}$ y MAPE, al igual que el modelo LSTM. Su desempeño también puede deberse a la naturaleza de los datos y a la falta de una configuración de parámetros avanzada del modelo.

Tabla 8

Métricas de rendimiento de todos los modelos evaluados para el Total de ventas diarias y los 25 productos más vendidos diariamente de la Cevichería Rico Norte.

	MAE				MSE				R2				MAPE			
	SARIMA	XGBOOST	LSTM	MLP	SARIMA	XGBOOST	LSTM	MLP	SARIMA	XGBOOST	LSTM	MLP	SARIMA	XGBOOST	LSTM	MLP
Total	421.946	133.455	671.573	675.807	378248	39788.6	868874	788971	0.61682	0.92931	-0.0235	0.07065	47.09%	15.95%	48.26%	39.16%
producto_16 (Combinado de 14)	6.09231	1.88803	8.71089	8.87503	78.3198	11.1353	171.578	143.865	0.43813	0.92335	-0.1499	0.03584	118.39%	47.67%	89.16%	85.22%
producto_17 (Combinado de 18)	2.79118	0.83937	4.18638	4.58136	18.291	2.13797	39.8736	32.9164	0.51356	0.92915	-0.0109	0.16546	104.08%	26.94%	68.20%	75.39%
producto_26 (Dúo Clásico)	2.92521	0.55165	4.02555	4.83669	21.7036	1.67062	44.6865	46.6019	0.34378	0.94581	-0.1949	-0.2462	63.49%	118.29%	165.72%	145.14%
producto_21 (Triple Tradicional)	3.46696	0.52329	3.84066	4.54885	33.009	1.10655	55.7136	51.7551	0.18677	0.9658	-0.1894	-0.1049	164.88%	88.49%	118.71%	124.57%
producto_49 (Gaseosa 1 Litro)	1.42158	0.35379	2.15524	2.45687	4.18505	0.28554	7.56659	9.50824	0.55614	0.95497	0.04491	-0.2002	68.63%	12.61%	55.21%	65.01%
producto_54 (Gaseosa Personal)	1.56828	0.38161	2.11445	2.28754	4.60563	0.41157	8.88836	8.25439	0.40647	0.95251	-0.1262	-0.0459	79.27%	51.36%	82.71%	84.89%
producto_2 (Ceviche de Pescado)	1.14629	0.31081	1.71733	1.70174	2.65903	0.24792	5.26477	4.8426	0.48028	0.94497	-0.1815	-0.0831	68.93%	21.38%	59.97%	58.69%
producto_45 (Leche de Tigre de 5)	1.2389	0.22937	1.6675	1.98419	2.82019	0.12317	4.68352	6.4479	0.39859	0.9687	-0.1586	-0.595	66.81%	18.60%	61.32%	78.49%

	MAE				MSE				R2				MAPE			
producto_57 (Cerveza Cuzqueña)	0.9455	0.19695	1.47194	1.89274	1.69369	0.10866	3.88795	5.60802	0.47467	0.96167	-0.3467	-0.9424	54.25%	22.13%	71.29%	85.78%
producto_48 (Jarra de Chicha)	1.04522	0.24107	1.43828	1.40275	2.31828	0.23904	4.36224	3.32469	0.31747	0.91452	-0.3021	0.00757	57.26%	33.24%	80.43%	72.78%
producto_53 (Gaseosa Gordita)	1.0308	0.18924	1.3995	1.43776	1.98066	0.14546	4.39721	3.44671	0.38624	0.94658	-0.4262	-0.1245	62.11%	29.11%	68.43%	68.77%
producto_31 (Chicharrón de Pota-Porción)	0.95568	0.20761	1.58395	1.52952	1.80951	0.13354	5.4482	3.72793	0.53025	0.95057	-0.4715	-0.0069	60.06%	34.35%	72.44%	69.76%
producto_52 (Gaseosa medio Litro)	0.92991	0.24464	1.19027	1.29338	2.02629	0.22815	4.02901	3.28525	0.35533	0.91697	-0.2643	-0.0309	57.38%	37.89%	71.15%	75.13%
producto_51 (Gaseosa 1 Litro y medio)	0.75444	0.16826	1.26501	1.66333	1.21126	0.10598	3.64191	4.44148	0.54228	0.94987	-0.5077	-0.8275	49.28%	29.93%	64.92%	88.60%
producto_55 (Chicha 2 Litros)	0.55639	0.19176	0.67681	0.85057	0.75751	0.15477	1.59979	1.43203	0.48994	0.92947	-0.1125	0.00416	38.89%	34.20%	48.32%	57.97%
producto_7 (Ceviche Clásico + Chicharrón de Pescado, Pota o Mixto)	0.6031	0.11635	0.78143	0.89377	0.72427	0.0444	1.46026	1.32372	0.46742	0.96945	-0.1298	-0.0242	43.21%	41.03%	66.42%	72.94%
producto_46 (Leche de Tigre de 12)	0.53802	0.12828	0.81361	1.1424	0.59057	0.05323	1.35196	2.12918	0.52302	0.95429	-0.1955	-0.8685	34.30%	52.18%	70.37%	94.72%
producto_37 (Chaufa Tradicional + Chicharrón de Pescado o Mixto)	0.50222	0.10369	0.57523	0.85784	0.53856	0.02772	0.85403	1.36792	0.38254	0.9637	-0.0478	-0.6782	34.76%	42.41%	59.79%	84.46%
producto_27 (Ronda	0.50966	0.06852	0.61946	0.66975	0.50063	0.01907	0.80389	0.79435	0.42	0.96272	-0.0865	0.01972	36.60%	31.39%	62.69%	64.39%

	MAE				MSE				R2				MAPE			
Marina (1))																
producto_28 (Ronda Marina (2))	0.49776	0.11842	0.60554	0.93807	0.54395	0.04018	0.87999	1.37054	0.39974	0.96538	-0.0456	-0.6285	36.30%	60.46%	62.52%	88.08%
producto_56 (Chicha Personal)	0.60216	0.09558	0.71613	0.79476	0.76055	0.04855	1.28295	1.25099	0.38567	0.95646	-0.0364	-0.0106	40.16%	47.05%	73.27%	77.86%
producto_22 (Triple a la Crema o Macho)	0.4726	0.07012	0.60991	0.88884	0.46061	0.03051	0.9561	1.34793	0.51412	0.95465	-0.0914	-0.5387	34.69%	33.88%	57.71%	89.11%
producto_38 (Chaufa Tradicional + Chicharrón de Pota)	0.48054	0.08238	0.53172	0.71975	0.45087	0.02777	0.65671	0.84616	0.35376	0.94781	-0.0081	-0.2989	35.23%	49.76%	64.41%	92.76%
producto_5 (Ceviche de Pescado + Chicharrón de Pescado, Pota o Mixto)	0.41834	0.06894	0.53551	0.77909	0.33151	0.01715	0.55144	0.92108	0.27449	0.96923	-0.174	-0.9609	34.83%	50.46%	89.86%	120.25%
producto_18 (Combinado con Mariscos)	0.60875	0.13408	0.75938	0.81786	1.13556	0.19401	2.72439	2.13472	0.35711	0.89098	-0.2672	0.00707	46.07%	143.98%	199.85%	161.83%

Nota: Elaboración propia

Salvo el inconveniente del MAPE, XGBoost viene a ser el mejor modelo de los 4 probados para nuestras predicciones.

Sobre el modelo XGBoost, lo que puede estar afectando al MAPE es la naturaleza de los datos, dado que los datos son números bajos cercanos a 0, y que en varias variables el 0 predomina más. Es por eso que, debido a los datos, es mejor enfocarse en las métricas MAE, MSE y $\boldsymbol{R^2}$, que no son alteradas por la presencia de ceros al ser métricas absolutas.

Otra razón puede deberse a la configuración de los parámetros del modelo.

```
model= xgb.XGBRegressor(
        objective='reg:squarederror',
        alpha= 0,
        gamma= 0.5,
        colsample_bytree= 1,
        subsample= 1,
        learning_rate= 0.3,
        n_estimators=300,
        seed=42,
        max_depth= 8)

model.fit(X_train, y_train)
```

Los parámetros presentados deben ser modificados para cada variable, mediante prueba y error o encontrar la combinación óptima de estos mediante una búsqueda avanzada llamada *GridSearchCV*; la desventaja de esta búsqueda es que requiere un alto coste computacional para realizarse con la cantidad de datos que actualmente se posee.

4.1.6. Despliegue

El sistema informático se desarrolló usando las tecnologías HTML, CSS y JavaScript para el frontend, y Python (Flask) y SQL para el backend, así como otras librerías para los gráficos estadísticos y los estilos.

Posterior a su desarrollo, se implementó el modelo XGBoost a dicho sistema.

4.2. RESULTADOS DEL MODELO DE PREDICCIÓN

Las comparaciones realizadas mediante las métricas de rendimiento entre los modelos probados destacaron a XGBoost, obteniendo un MAE, MSE y R^2 mucho más óptimos que sus competidores.

Como un modo adicional para evaluar la precisión del modelo, se aplicó la prueba t-Student de dos muestras para datos emparejados. Esta prueba se realizó para comparar las predicciones del modelo XGBoost con los datos reales de las ventas desde el 29 de febrero del 2024 hasta el 27 de abril del 2024.

Tabla 9

Prueba t-Student entre los datos predichos y los datos reales de las ventas diarias desde el 29/02/2024 hasta el 27/04/2024.

Variable	Estadístico t	Valor p
Total	-0.289359752	0.772935326
producto_16	2.568324909	0.012726927
producto_17	0.222348251	0.824636872
producto_26	-1.600671494	0.112947033
producto_21	-0.147696363	0.882903544
producto_49	0.606739424	0.54553034
producto_54	-1.122397006	0.264498689
producto_2	-0.762636388	0.447562661
producto_45	0.75493102	0.452190434
producto_57	-0.60764362	0.54514357
producto_48	1.207245904	0.232459508
producto_53	-1.095254983	0.276199693
producto_31	2.070762744	0.041566468
producto_52	0.732333837	0.466315537
producto_51	1.333882369	0.18546661
producto_55	0.574024444	0.567342282
producto_7	3.225183075	0.001728176
producto_46	3.647932553	0.000465985
producto_37	2.372684056	0.020016889
producto_27	2.848110223	0.005543366
producto_28	2.149493162	0.03410942
producto_56	1.576810368	0.118947304
producto_22	3.771485315	0.0003085
producto_38	3.012529228	0.0033712
producto_5	0.576849749	0.565529714
producto_18	0.40596412	0.685708815

Nota: Elaboración propia

Se aplicó un nivel de significancia de $\alpha = 0.05$.

Se puede decir de esta comparación que en la mayoría de las variables no se presenta una diferencia significativa entre los datos pronosticados y los datos reales.

Las variables *producto_16, producto_31, producto_7, producto_46, producto_37, producto_27, producto_28, producto_22 y producto_38* presentaron una diferencia significativa entre los datos pronosticados y los datos reales.

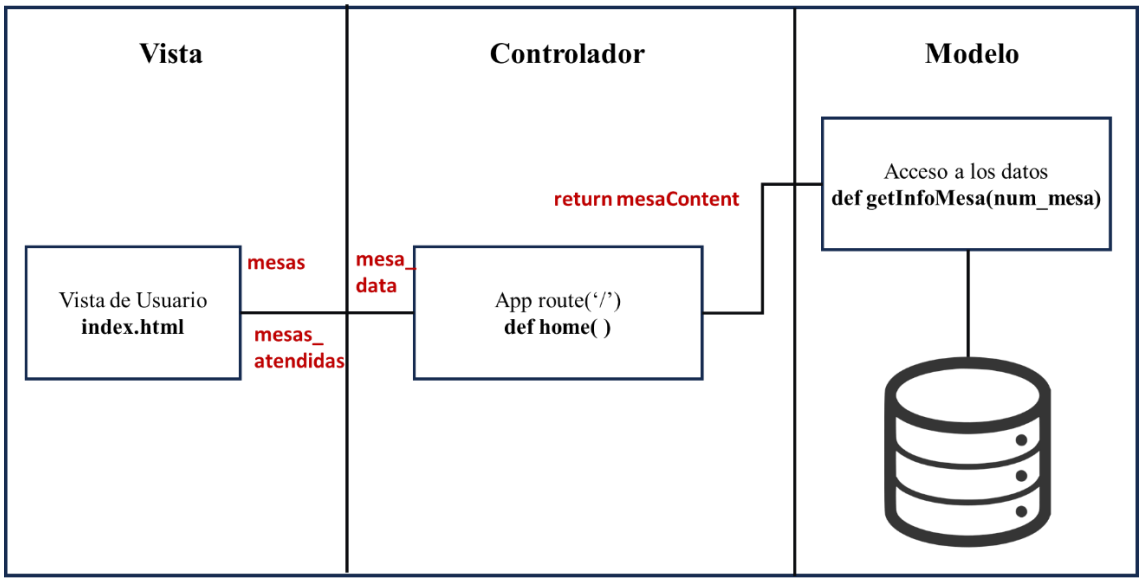
En el contexto de la prueba t-Student, se acepta la hipótesis alternativa para las variables que presentaron una diferencia significativa entre los datos pronosticados y los datos reales. Y se rechaza la hipótesis alternativa para las demás variables.

4.3. RESULTADOS DEL SISTEMA INFORMÁTICO

Para el desarrollo del sistema informático se empleó un modelo de desarrollo Modelo – Vista – Controlador (MVC) para su desarrollo, en el cual el framework de Python, FLask, calzó perfectamente.

Figura 34

Representación de MVC del sistema informático.

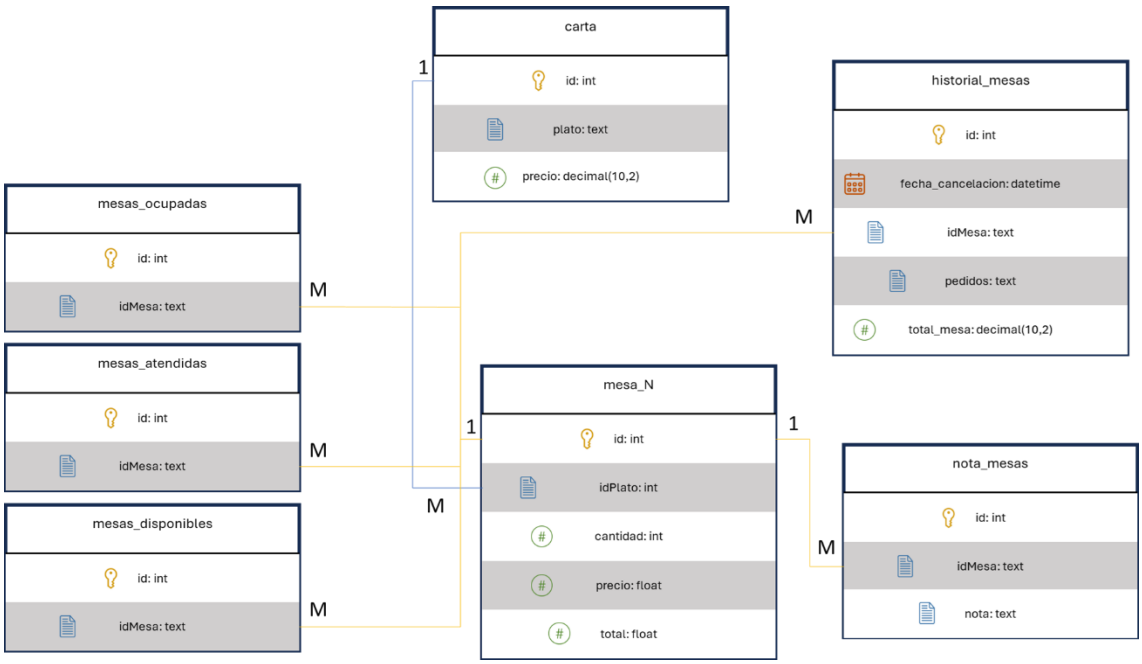


Nota: Elaboración propia

Se diseñó una base de datos relacional localmente con la ayuda del gestor phpmyadmin.

Figura 35

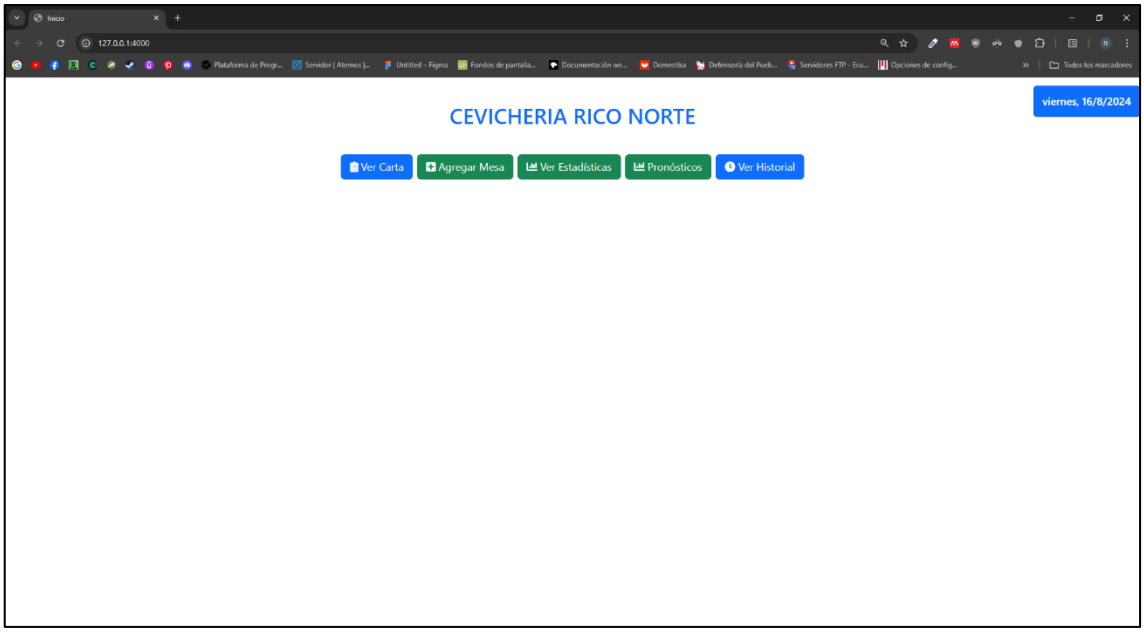
Estructura de la base de datos del sistema informático.



Nota: Elaboración propia

Figura 36

Página principal del sistema.



Nota: Interfaz de la página principal

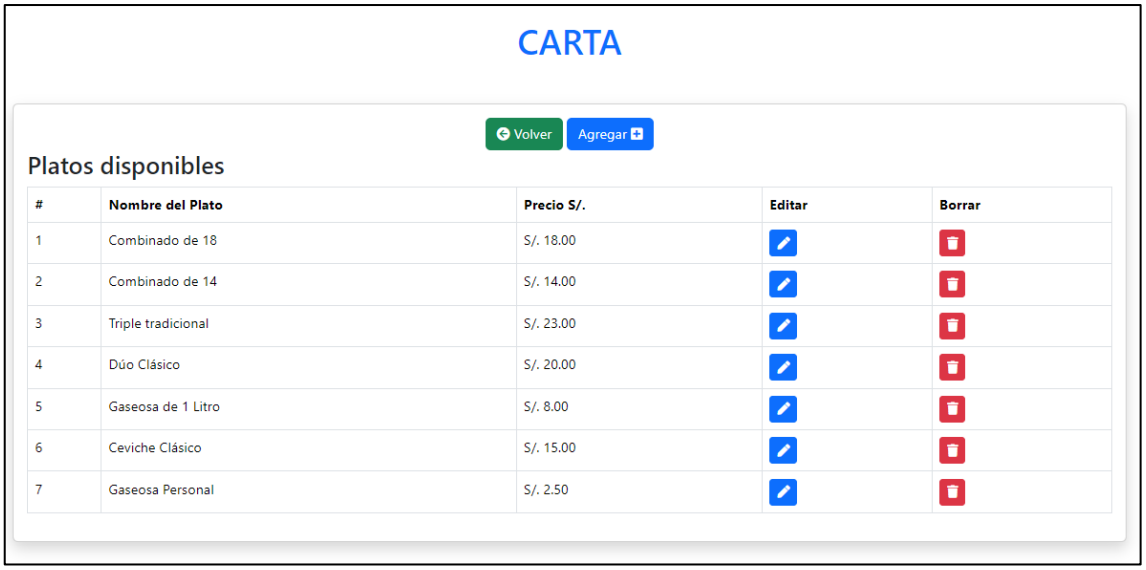
El sistema informático inteligente está conformado por 6 apartados principales:

- Ver Carta
- Agregar Mesa
- Mesa_X
- Ver Historial
- Ver Estadísticas
- Pronósticos

El apartado “Ver Carta” permite ver, agregar, editar o borrar los productos que se ofrecen en la cevichería Rico Norte.

Figura 37

Interfaz del apartado" Ver Carta".

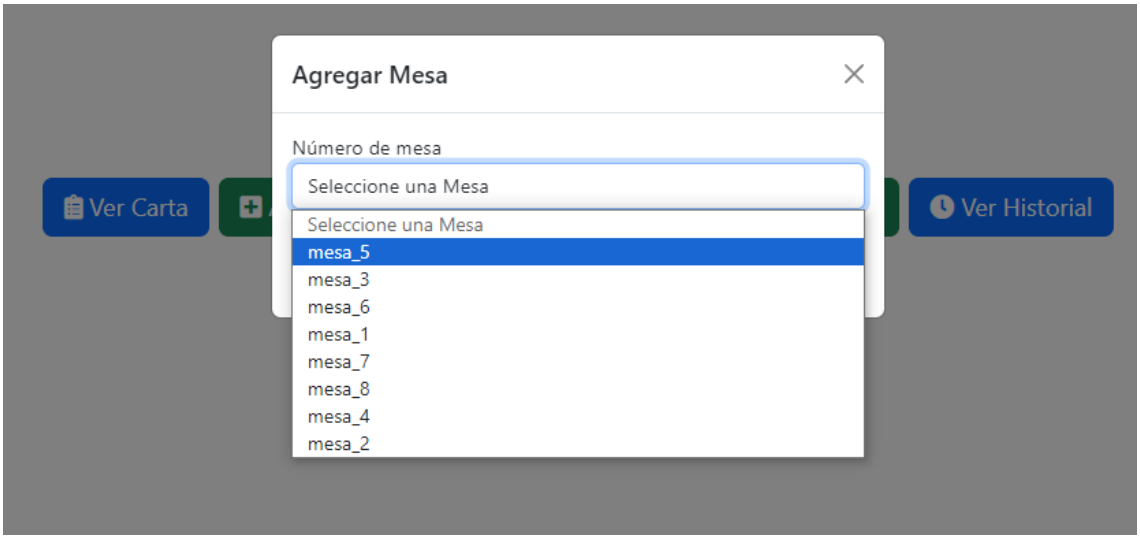


Nota: Interfaz de la Carta

El apartado “Agregar Mesa” permite seleccionar y agregar el número de mesa en donde se sentó el comensal, dicha mesa servirá de tabla para agregar los pedidos que se realicen.

Figura 38

Formulario para añadir una mesa.



Nota: Formulario de Agregar Mesa

Se añadieron 3 mesa de ejemplo.

Figura 39

Mesas añadidas.

Nota: Mesas Agregadas en la Página Principal

El apartado “Mesa_X” viene a ser la interfaz propia de cada mesa que se añadió. En dicha interfaz se pueden ver, agregar, editar o borrar los pedidos que ordenó el comensal sentado en esa mesa.

La interfaz de cada mesa posee un cuadro de texto para añadir notas adicionales con respecto a los pedidos del comensal, también posee la opción de “Descartar Mesa” en caso de que el comensal cancele su orden, y la opción de “Confirmar Pago”, la cual guardará en la base de datos la fecha, el número de mesa y los pedidos realizados en ella, junto con la nota adicional.

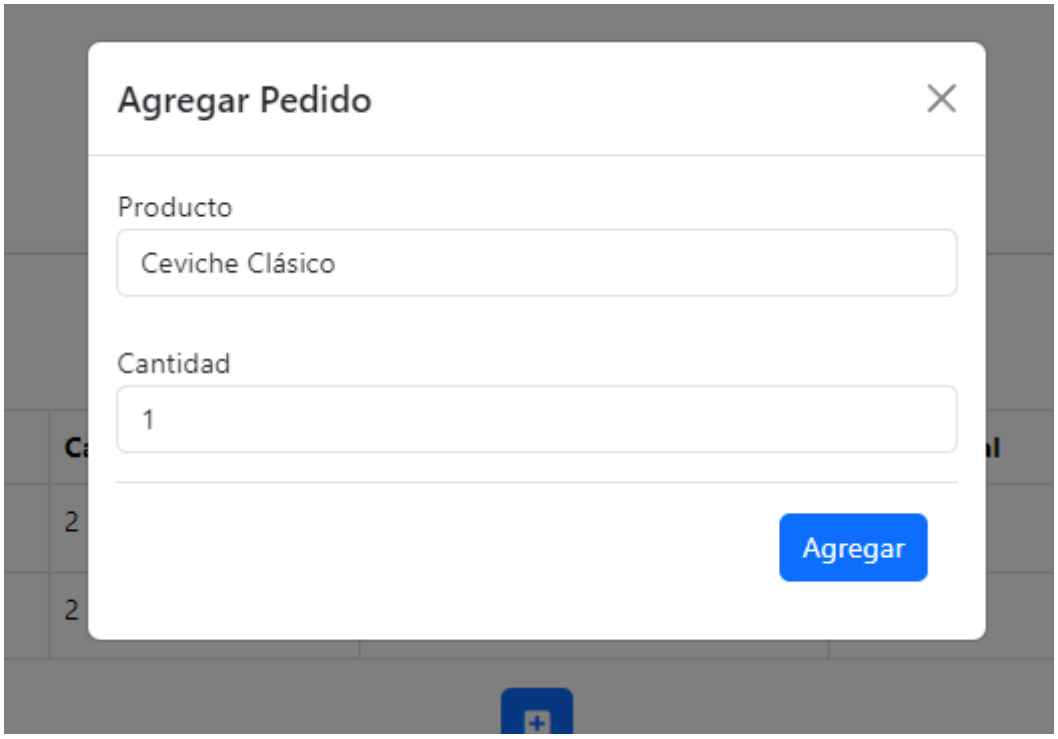
Figura 40

Interfaz de la Mesa 5.

Nota: Interfaz de la Mesa 5 con productos y nota agregados.

Figura 41

Formulario para agregar un pedido a la mesa.



Nota: Agregando un pedido a la Mesa 5.

Las mesas añadidas que se visualizan en la página principal poseen un botón para acceder a la interfaz de cada mesa, y un botón para marcar que esa mesa ya fue atendida.

Figura 42

Ejemplo de mesa atendida.

Nota: Marcando una mesa como Atendida.

El apartado “Ver Historial” permite ver las mesas con sus respectivos pedidos que fueron guardadas en la base de datos. En la interfaz de este apartado figura un formulario de calendario que permite elegir una fecha específica.

Figura 43

Formulario de calendario de la interfaz "Ver Historial".

Nota: Seleccionando una fecha en el formulario de calendario.

Figura 44

Historial de mesas de la fecha seleccionada.

Nota: Historial de venta de la fecha seleccionada.

El apartado “Ver Estadísticas” permite ver datos estadísticos relevantes sobre las ventas de un día, semana o mes en específico. Al igual que el apartado anterior, posee un formulario de calendario diario, semanal y mensual.

Figura 45

Estadísticas de la fecha seleccionada.

Nota: Estadísticas de venta de la fecha seleccionada.

El apartado “Pronósticos”, que funciona gracias al script del modelo *XGBoost* implementado, presenta una interfaz que muestra una tabla con los pronósticos, algunas estadísticas relevantes, y un formulario de calendario para poder seleccionar una fecha de inicio y fin para el pronóstico.

El script de XGBoost fue modificado de la siguiente manera para que pueda ser usado en este apartado:

```
def pronosticar(fecha_inicio, fecha_fin):
    #Cargar datos
    df =
pd.read_csv('C:/Users/Rodrigo/Desktop/SISTEMA_CEVICHERIA/ventasCev_mod
ificado.csv', delimiter=',')

variables = ['Total', 'producto_16', 'producto_17', 'producto_26',
'producto_21', 'producto_49', 'producto_54', 'producto_2',
'producto_45', 'producto_57', 'producto_48', 'producto_53',
'producto_31', 'producto_52', 'producto_51', 'producto_55',
```

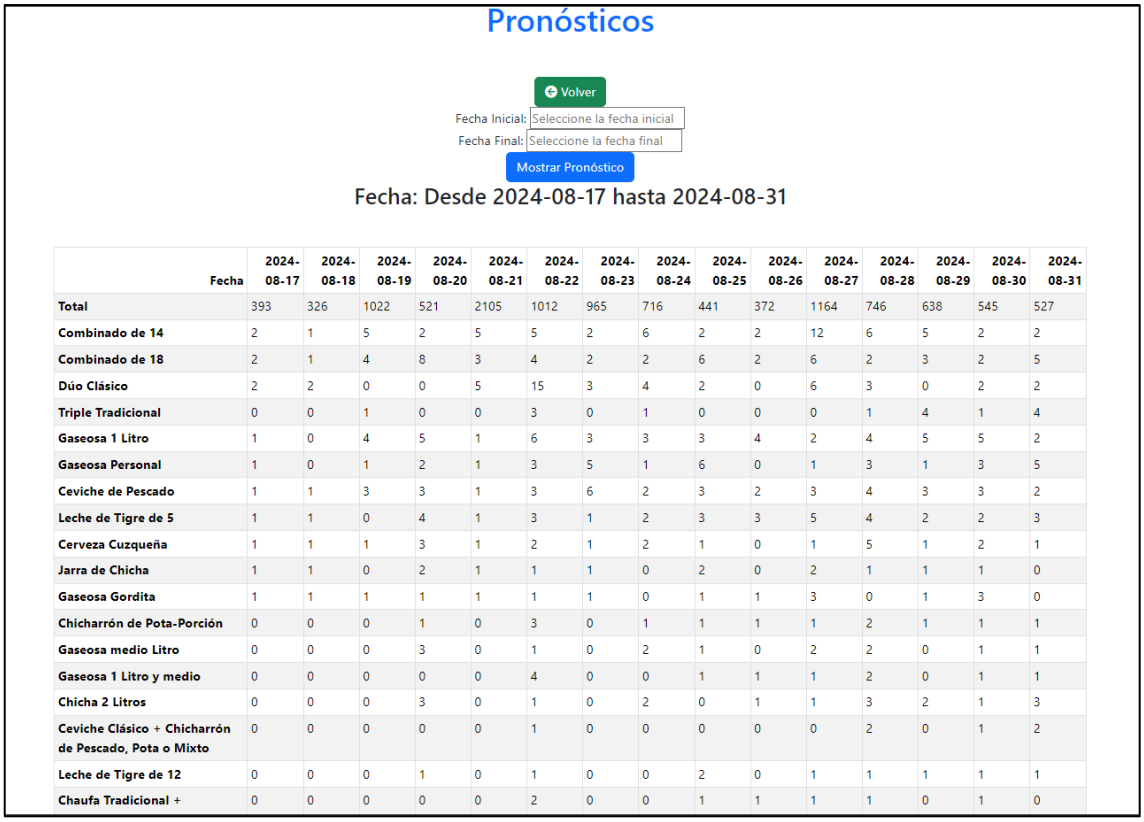
```
'producto_7', 'producto_46', 'producto_37', 'producto_27',
'producto_28', 'producto_56', 'producto_22', 'producto_38',
'producto_5', 'producto_18']

    #Cambiar formato de las fechas
    fecha_inicio= datetime.strptime(fecha_inicio, '%Y-%m-%d')
    fecha_fin= datetime.strptime(fecha_fin, '%Y-%m-%d')

    #Obtener periodo de días
    periods= fecha_fin - fecha_inicio
    periods= periods.days

    #Convertir la columna Fecha a datetime
    df['Fecha'] = pd.to_datetime(df['Fecha'])

    #Características adicionales
    df['Dia_de_semana'] = df['Fecha'].dt.dayofweek
    df['Dia_del_mes'] = df['Fecha'].dt.day
    df['Dia_del_año'] = df['Fecha'].dt.dayofyear
    df['Año'] = df['Fecha'].dt.year
    df['Mes'] = df['Fecha'].dt.month
    df['Es_fin_de_semana'] = df['Dia_de_semana'].isin([5,
6]).astype(int)
    df['Semana_del_año'] = df['Fecha'].dt.isocalendar().week
    df['Trimestre'] = df['Fecha'].dt.quarter

    #Fechas futuras
    dias_futuros = pd.date_range(start=fecha_inicio, periods=periods +
1)
    futuro_df = pd.DataFrame(dias_futuros, columns=['Fecha'])

    #Características para fechas futuras
    futuro_df['Dia_del_año'] = futuro_df['Fecha'].dt.dayofyear
    futuro_df['Año'] = futuro_df['Fecha'].dt.year
    futuro_df['Mes'] = futuro_df['Fecha'].dt.month
    futuro_df['Dia_de_semana'] = futuro_df['Fecha'].dt.dayofweek
    futuro_df['Es_fin_de_semana'] =
futuro_df['Dia_de_semana'].isin([5, 6]).astype(int)

    futuro_df['Dia_del_mes'] = df['Fecha'].dt.day
    futuro_df['Semana_del_año'] = df['Fecha'].dt.isocalendar().week
    futuro_df['Trimestre'] = df['Fecha'].dt.quarter

    predicciones= pd.DataFrame(dias_futuros, columns=['Fecha'])

    for variable in variables:
        #Calcular Z-scores
        z_scores = stats.zscore(df[variable])
```

```
            #Eliminar outliers
            df= df[(z_scores >= -3) & (z_scores <= 3)]

            df['Lag_1'] = df[variable].shift(1)
            df['Lag_7'] = df[variable].shift(7)

            #Medias móviles
            df['Media_movil_3'] = df[variable].rolling(window=3).mean()
            df['Media_movil_7'] = df[variable].rolling(window=7).mean()
            df['Media_movil_30'] = df[variable].rolling(window=30).mean()
            df['Diferencia_dia_anterior'] = df[variable].diff()

            #df.dropna(inplace=True)  #Eliminar filas con valores NaN

            #Variables predictoras
           X = df[['Dia_del_año', 'Año', 'Mes', 'Dia_de_semana',
          'Es_fin_de_semana', 'Dia_del_mes', 'Semana_del_año',
'Trimestre', 'Lag_1', 'Lag_7', 'Media_movil_3', 'Media_movil_7',
'Media_movil_30', 'Diferencia_dia_anterior']]

            #Variable objetivo
            y = df[variable]

            #División en conjuntos de entrenamiento y prueba
            X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

            #Crear y entrenar el modelo para la predicción de "y"
            if variable != 'Total':
                model= xgb.XGBRegressor(
                    objective='reg:squarederror',
                    alpha= 0,
                    gamma= 0.5,
                    colsample_bytree= 1,
                    subsample= 1,
                    learning_rate= 0.01,
                    n_estimators=300,
                    seed=42,
                    max_depth= 3)

            else:
                model= xgb.XGBRegressor(
                    objective='reg:squarederror',
                    alpha= 0,
                    colsample_bytree= 1.0,
                    subsample= 0.8,
                    learning_rate= 0.01,
                    n_estimators=300,
                    seed=42,
```

```
                    max_depth= 6)

            model.fit(X_train, y_train)

            futuro_df['Media_movil_3'] =
df[variable].rolling(window=3).mean()
            futuro_df['Media_movil_7'] =
df[variable].rolling(window=7).mean()
            futuro_df['Media_movil_30'] =
df[variable].rolling(window=30).mean()
            futuro_df['Diferencia_dia_anterior'] = df[variable].diff()

            last_known_values = df[['Fecha',
variable]].sort_values(by='Fecha').tail(7)
            futuro_df = futuro_df.merge(last_known_values, how='left',
on='Fecha')

            futuro_df['Lag_1'] = futuro_df[variable].shift(1)
            futuro_df['Lag_7'] = futuro_df[variable].shift(7)
            futuro_df.fillna(0, inplace=True)  #Rellenar NaNs con 0

            #Hacer predicciones
            X_futuro = futuro_df[['Dia_del_año', 'Año', 'Mes',
'Dia_de_semana', 'Es_fin_de_semana', 'Dia_del_mes', 'Semana_del_año',
'Trimestre', 'Lag_1', 'Lag_7', 'Media_movil_3', 'Media_movil_7',
'Media_movil_30', 'Diferencia_dia_anterior']]
            pred_futuro = model.predict(X_futuro)

            #Agregar predicciones al DataFrame
            predicciones[variable] = np.round(pred_futuro).astype(int)

    print(predicciones)
    return predicciones
```

Figura 46

Pronósticos para las fechas seleccionadas.

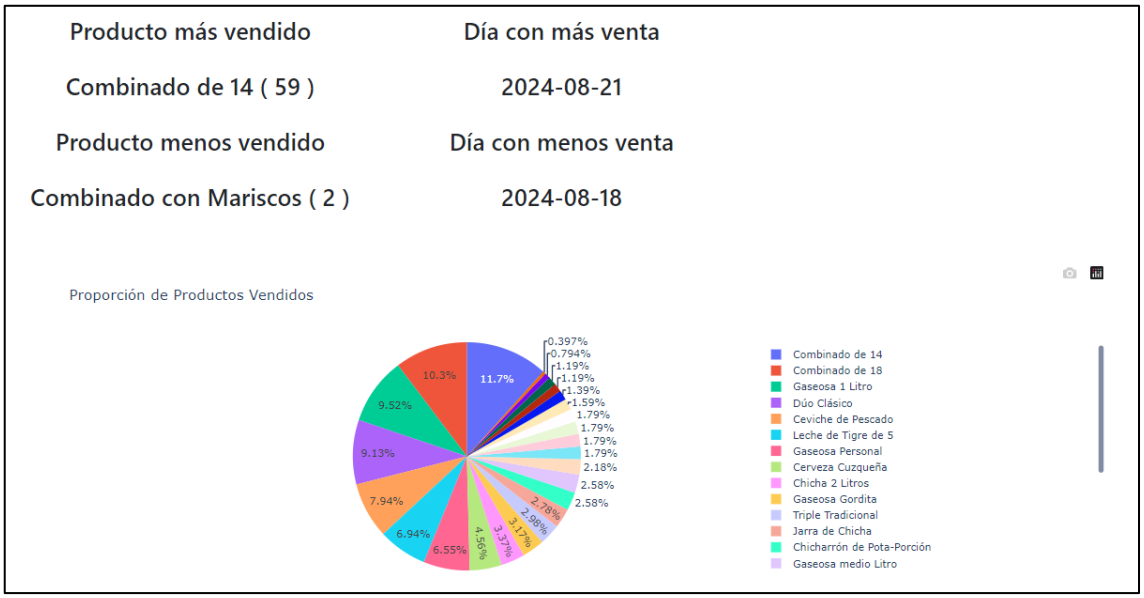


Fecha	2024-08-17	2024-08-18	2024-08-19	2024-08-20	2024-08-21	2024-08-22	2024-08-23	2024-08-24	2024-08-25	2024-08-26	2024-08-27	2024-08-28	2024-08-29	2024-08-30	2024-08-31
Total	393	326	1022	521	2105	1012	965	716	441	372	1164	746	638	545	527
Combinado de 14	2	1	5	2	5	5	2	6	2	2	12	6	5	2	2
Combinado de 18	2	1	4	8	3	4	2	2	6	2	6	2	3	2	5
Dúo Clásico	2	2	0	0	5	15	3	4	2	0	6	3	0	2	2
Triple Tradicional	0	0	1	0	0	3	0	1	0	0	0	1	4	1	4
Gaseosa 1 Litro	1	0	4	5	1	6	3	3	3	4	2	4	5	5	2
Gaseosa Personal	1	0	1	2	1	3	5	1	6	0	1	3	1	3	5
Ceviche de Pescado	1	1	3	3	1	3	6	2	3	2	3	4	3	3	2
Leche de Tigre de 5	1	1	0	4	1	3	1	2	3	3	5	4	2	2	3
Cerveza Cuzqueña	1	1	1	3	1	2	1	2	1	0	1	5	1	2	1
Jarra de Chicha	1	1	0	2	1	1	1	0	2	0	2	1	1	1	0
Gaseosa Gordita	1	1	1	1	1	1	1	0	1	1	3	0	1	3	0
Chicharrón de Pota-Porción	0	0	0	1	0	3	0	1	1	1	1	2	1	1	1
Gaseosa medio Litro	0	0	0	3	0	1	0	2	1	0	2	2	0	1	1
Gaseosa 1 Litro y medio	0	0	0	0	0	4	0	0	1	1	1	2	0	1	1
Chicha 2 Litros	0	0	0	3	0	1	0	2	0	1	1	3	2	1	3
Ceviche Clásico + Chicharrón de Pescado, Pota o Mixto	0	0	0	0	0	1	0	0	0	0	0	2	0	1	2
Leche de Tigre de 12	0	0	0	1	0	1	0	0	2	0	1	1	1	1	1
Chaufa Tradicional +	0	0	0	0	0	2	0	0	1	1	1	1	0	1	0

Nota: Pronóstico generado para los días especificados.

Figura 47

Estadísticas de los datos pronosticados.



Nota: Estadísticas y gráfico generados para los días especificados.

4.4. DISCUSIÓN

Según los resultados obtenidos por las métricas MAE, MSE y $\boldsymbol{R^2}$, se acepta la hipótesis que señala que la implementación de un sistema informático inteligente logró predecir con un mínimo margen de error las ventas de la Cevichería Rico Norte de la ciudad de Puno.

Según los resultados obtenidos por la prueba t-Student, se rechaza la hipótesis que señala que la implementación de un sistema informático inteligente logró predecir con un mínimo margen de error las ventas de la Cevichería Rico Norte de la ciudad de Puno para las variables *producto_16, producto_31, producto_7, producto_46, producto_37, producto_27, producto_28, producto_22 y producto_38*. Y aceptamos nuestra hipótesis para las demás variables.

Referido al preprocesamiento de los datos, eliminar los valores atípicos y crear características temporales basándose en los datos actuales ayudó al modelo a mejorar sus métricas de rendimiento.

Referido al modelo de predicción, XGBoost fue el modelo que sacó los mejores puntajes en las métricas de rendimiento, compartiendo el mismo resultado con el estudio de Yi (2023), donde el modelo XGBoost superó a otros modelos al momento de predecir las ventas de Walmart. El estudio de Merdas & Mousa (2023) también destaca como uno de los mejores algoritmos a XGBoost para predecir ventas de comida. Por último, Mohanapriya & Mohana Saranya (2020) vuelve a destacar a XGBoost en la predicción de ventas.

Referente al sistema informático inteligente, este facilita la creación y el acceso a las predicciones por parte del administrador, potenciando aún más el sistema informático

que, de por sí, ya es una herramienta crucial para mejorar en múltiples aspectos la gestión de la cevichería Rico Norte.

V. CONCLUSIONES

PRIMERA: La presente investigación logró implementar un sistema informático inteligente para el pronóstico de ventas de la cevichería Rico Norte de la ciudad de Puno, para lograrlo, se compararon 4 modelos de Machine Learning: SARIMA, XGBoost, LSTM (Long Short Term Memory) y MLP (Modelo de Perceptrón Multicapa), concluyendo que el modelo XGBoost fue el mejor de los 4 según los resultados que obtuvo en las métricas de rendimiento MAE, MSE y $\boldsymbol{R^2}$, mostrando un MAE y MSE bajos en comparación a los otros modelos, y siendo el único modelo que obtuvo buenos porcentajes de $\boldsymbol{R^2}$ (explicando más del 90% de la variabilidad de los datos en todas las variables), demostrando así ser eficiente para predecir futuras ventas de la cevichería Rico Norte.

SEGUNDA: Se recopilaron los datos necesarios de forma manual proporcionados por el dueño, los cuales se transcribieron a una tabla de Excel que posteriormente se trasformó en formato CSV. Estos datos fueron preprocesados mediante un script de Python, seleccionando los 25 productos más vendidos, eliminando los valores atípicos y creando características temporales para mejorar los resultados del modelo.

TERCERA: Se lograron implementar los modelos de Machine Learning SARIMA, XGBoost, LSTM y MLP a los datos preprocesados mediante scripts de Python, apoyándose de librerías como scikitlearn, tensorflow, scipy, entre otras.

CUARTA: Se evaluaron los modelos usando las siguientes métricas de rendimiento: Error Absoluto Medio (MAE), Error Cuadrático Medio (MSE), R-

cuadrado ($\boldsymbol{R^2}$) y Error Porcentual Absoluto Medio (MAPE). El modelo que obtuvo mejores resultados en MAE, MSE Y $\boldsymbol{R^2}$ fue XGBoost, excepto en MAPE, que, debido a la naturaleza de los datos, en casi todas las variables estas tienden a 0.

QUINTA: Se logró diseñar y codificar el sistema informático inteligente usando las tecnologías HTML, CSS, JavaScript, Python (Flask) y SQL, mediante el modelo de desarrollo Modelo – Vista – Controlador (MVC), el cual posee 6 apartados: Ver Carta, Agregar Mesa, Mesa_X, Ver Historial, Ver Estadísticas y Pronósticos, cada uno con funcionalidades útiles para el administrador de la cevichería Rico Norte. Se destaca el apartado de Pronósticos, que tiene integrado al script del modelo de predicción XGBoost. Dicho apartado facilita la creación de pronósticos para el administrador.

VI. RECOMENDACIONES

PRIMERA: Se recomienda usar el modelo XGBoost para investigaciones relacionadas con la predicción de ventas en comida, puesto que este modelo obtuvo los mejores resultados con respecto a los otros al momento de realizar las predicciones.

SEGUNDA: Con respecto a la métrica MAPE obtenida por XGBoost, se recomienda realizar la búsqueda avanzada *GridSearchCV* proporcionada por la librería scikitlearn de Python, pero solo si se tiene la potencia computacional suficiente y los datos no son muy extensos.

TERCERA: Si se desea usar otro tipo de modelos, como LSTM o MLP, se recomienda usar otros métodos de preprocesamiento de datos y cambiar los parámetros de los modelos hasta encontrar los valores óptimos.

CUARTA: Se recomienda eliminar los valores atípicos y crear características temporales en el preprocesamiento de datos para un mejor desempeño de los modelos.

QUINTA: Si se desea hacer un sistema informático similar, se recomienda usar la librería Flask de Python como backend por su fácil curva de aprendizaje y el buen encaje que tiene al modelo de desarrollo Modelo – Vista – Controlar (MVC), aparte de que es más sencillo integrar gráficos y modelos de predicción debido a que el entorno Python proporciona todo lo anterior mediante su gran número de librerías.

VII. REFERENCIAS BIBLIOGRÁFICAS

Abbott, D. (2014). *Applied Predictive Analytics*.

Aggarwal, C. C. (2018). *Neural Networks and Deep Learning*. https://doi.org/10.1007/978-3-319-94463-0

Alpaydin, E. (2014). *Introduction to Machine Learning* (3rd ed.).

AnalyticsVidhya. (2024). *What is XGBoost Algorithm?* AnalyticsVidhya. https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/

Andrade León, J. R. (2022). *La importancia del pronóstico en la empresa*. Linkedin. https://www.linkedin.com/pulse/la-importancia-del-pron%C3%B3stico-en-empresa-jes%C3%BAs-rodolfo/

AWS. (2022). *¿Qué es el aprendizaje automático?* AWS. https://aws.amazon.com/es/what-is/machine-learning/

AWS. (2024a). *¿Qué es el boosting?* AWS. https://aws.amazon.com/es/what-is/boosting/

AWS. (2024b). *¿Qué es JavaScript (JS)?* AWS. https://aws.amazon.com/es/what-is/javascript/

AWS. (2024c). *¿Qué es Python?* AWS. https://aws.amazon.com/es/what-is/python/

AWS. (2024d). *¿Qué es SQL (lenguaje de consulta estructurada)?* AWS. https://aws.amazon.com/es/what-is/sql/

AWS. (2024e). *¿Qué es una RNN (red neuronal recurrente)?* AWS. https://aws.amazon.com/es/what-is/recurrent-neural-network/#:~:text=Una%20red%20neuronal%20recurrente%20(RNN,salida%20de%20datos%20secuencial%20espec%C3%ADfica.

Baena Paz, G. (2017). *Metodología de la Investigación* (3rd ed.).

BookDown. (2022). *Capítulo 8 Análisis de Series de Tiempo*. BookDown. https://bookdown.org/keilor_rojas/CienciaDatos/an%C3%A1lisis-de-series-de-tiempo.html

Castañeda Rojas, M. A. D. (2020). *Machine Learning Para La Gestión De Ventas En La Empresa Vértice Empresarial S.A.C* [Universidad César Vallejo]. https://hdl.handle.net/20.500.12692/56129

Chen, P., Niu, A., Liu, D., Jiang, W., & Ma, B. (2018). Time Series Forecasting of Temperatures using SARIMA: An Example from Nanjing. *IOP Conference Series: Materials Science and Engineering*, *394*(5). https://doi.org/10.1088/1757-899X/394/5/052024

Cheng, J., Dong, L., & Lapata, M. (2016). *Long Short-Term Memory-Networks for Machine Reading*. http://arxiv.org/abs/1601.06733

Clavijo, C. (2023). *Cómo hacer un pronóstico de ventas: tipos, pasos y ejemplos.* HubSpot. https://blog.hubspot.es/sales/pronostico-de-ventas

DataBitAI. (2023). *Métricas de Evaluación en Machine Learning.* DataBitAI. https://databitai.com/machine-learning/metricas-de-evaluacion-en-machine-learning/

Datision. (2021, August 5). *10 aplicaciones de Machine Learning en los negocios*. Datision. https://datision.com/blog/aplicaciones-machine-learning-negocios/

Domingo Muñoz, J. (2017, November 17). *Qué es Flask*. OpenWebinars. https://openwebinars.net/blog/que-es-flask/#:~:text=Te%20contamos%20qu%C3%A9%20es%20Flask,con%20pocas%20lineas%20de%20c%C3%B3digo.&text=En%20la%20actualidad%20existen%20muchas,sencilla%20aplicaciones%20web%20con%20Python

Elmasri, R., & Navathe, S. B. (2011). *Fundamentals of Database Systems, Sixth Edition*.

Escala. (2024). *¿Qué es la automatización de negocios y cómo hacerlo?* Escala. https://escala.com/automatizacion-de-negocios/

García Fernández, C. (2023, November 14). *Importancia del preprocesamiento de datos en proyectos de Data Science*. OpenWebinars. https://openwebinars.net/blog/importancia-del-preprocesamiento-de-datos-en-proyectos-de-data-science/

González Casimiro, M. P. (2009). *Análisis de series temporales: Modelos ARIMA*.

González, L. (2018, November 2). *Introducción a la librería Scikit-Learn de Python*. AprendeIA. https://aprendeia.com/libreria-scikit-learn-de-python/

González, L. (2021, October 5). *¿Qué es el Perceptrón? Perceptrón Simple y Multicapa*. AprendeIA. https://aprendeia.com/que-es-el-perceptron-simple-y-multicapa/#:~:text=Un%20modelo%20de%20perceptr%C3%B3n%20multicapa,y%20la%20etapa%20hacia%20atr%C3%A1s

González, L. (2024). *¿Qué es TensorFlow? ¿Cómo funciona?* AprendeIA. https://aprendeia.com/que-es-tensorflow-como-funciona/

Graves, A. (2012). Long Short-Term Memory. *Springer Link*, 37–45.

Hotz, N. (2024, April 28). *What is CRISP DM?* Data Science Process Alliance. https://www.datascience-pm.com/crisp-dm-2/

Hussam Mezher Merdas, & Ayad Hameed Mousa. (2023). Food Sales Prediction Using MLP, RANSAC, and Bagging. *Journal of Techniques*, *5*(4), 202–208. https://doi.org/10.51173/jt.v5i4.1458

IBM. (2021a). *Conceptos básicos de ayuda de CRISP-DM*. IBM. https://www.ibm.com/docs/es/spss-modeler/saas?topic=dm-crisp-help-overview

IBM. (2021b). *¿Qué es el machine learning (ML)?* IBM. https://www.ibm.com/es-es/topics/machine-learning

IBM. (2024). *¿Qué es el aprendizaje no supervisado?* IBM. https://www.ibm.com/es-es/topics/unsupervised-learning

Izquierdo Henríquez, M. I., Solano Coello, M. E., & Tapia Sánchez, C. D. (2020). Modelo de Pronóstico de la Exportación Minera en el Perú, 2020. *Revista Electrónica Universidad Nacional de Trujillo*. https://revistas.unitru.edu.pe/index.php/REDIES/article/view/4419/4833

Lenis, A. (2024). *Inteligencia artificial: qué es y las ventajas de usarla en tu empresa*. Hubspot. https://blog.hubspot.es/marketing/inteligencia-articial-esta-aqui

Martel Vicente, C. O. (2021). *Implementación de un sistema web para el control de las ventas del comedor restaurante el CREVAL en la ciudad de Huánuco – 2020* [Universidad de Huánuco]. http://repositorio.udh.edu.pe/123456789/3113

Matthes, E. (2019). *Python Crash Course* (R. Hoffman, Ed.; 2nd ed.). William Pollock. https://khwarizmi.org/wp-content/uploads/2021/04/Eric_Matthes_Python_Crash_Course_A_Hands.pdf

MDN. (2023). *HTML: Lenguaje de etiquetas de hipertexto*. MDN. https://developer.mozilla.org/es/docs/Web/HTML

MDN. (2024a). *CSS*. MDN. https://developer.mozilla.org/es/docs/Web/CSS

MDN. (2024b). *JavaScript*. MDN. https://developer.mozilla.org/es/docs/Web/JavaScript

Merdas, H. M., & Mousa, A. H. (2023). Food sales prediction model using machine learning techniques. *International Journal of Electrical and Computer Engineering*, *13*(6), 6578–6585. https://doi.org/10.11591/ijece.v13i6.pp6578-6585

Mills, T. C. (2019). *Applied Time Series Analysis* (S. Ikeda, Ed.). Candice Janco.

Mohanapriya, S., & Mohana Saranya, S. (2020). Sales prediction using machine learning algorithm. *International Journal of Advanced Science and Technology*, *29*(3 Special Issue), 1049–1055.

Moreno Pérez, J. Carlos., & Ramos Pérez, A. Francisco. (2014). *Administración hardware de un sistema informático*. Ra-Ma. https://www-digitaliapublishing-com.bibliotecavirtualunap.remotexs.co/a/109981

Mueller, J. P., & Massaron, L. (2019). *Deep Learning for Dummies*.

Naik, H., Yashwanth, K., Suraj, P., & Jayapandian, N. (2022). Machine Learning based Food Sales Prediction using Random Forest Regression. *6th International Conference on Electronics, Communication and Aerospace Technology, ICECA 2022 - Proceedings*, 998–1004. https://doi.org/10.1109/ICECA55336.2022.10009277

Navarro, S. (2024, April 18). *¿Para qué sirve TensorFlow?* KeepCoding. https://keepcoding.io/blog/para-que-sirve-tensorflow/

Naveen Joshi. (2023, February 21). *How Machine Learning Is Improving Restaurants*. BBNTimes. https://www.bbntimes.com/technology/how-machine-learning-is-improving-restaurants

Nazmuz Sakib, S. M. (2023). *Restaurant Sales Prediction Using Machine Learning*. https://doi.org/10.31224/osf.io/wa927

Newbold, P., Carlson, W. I., & Thorne, B. M. (2008). *Estadística para Administración y Economía*.

Ortega, C. (2024). *Herramientas de inteligencia artificial: 5 ejemplos y sus características*. QuestionPro. https://www.questionpro.com/blog/es/herramientas-de-inteligencia-artificial/

Pavlyshenko, B. M. (2019). Machine-learning models for sales time series forecasting. *Data*, *4*(1). https://doi.org/10.3390/data4010015

Ponce Illacutipa, G. M. (2022). *Modelo basado en Machine Learning para optimizar el pronóstico de ventas de la empresa Ricos Pan, año 2020 - 2021* [Universidad Nacional del Altiplano Puno]. https://repositorio.unap.edu.pe/handle/20.500.14082/19132

PyPi. (2024). *Flask 3.0.3*. PyPi. https://pypi.org/project/Flask/

Rohaan, D., Topan, E., & Groothuis-Oudshoorn, C. G. M. (2022). Using supervised machine learning for B2B sales forecasting: A case study of spare parts sales forecasting at an after-sales service provider. *Expert Systems with Applications*, *188*. https://doi.org/10.1016/j.eswa.2021.115925

Romero, N. J. (2022, April 6). *Pronóstico de ventas: ¿Por qué es importante?* Oduka. https://oduka.co/pronostico-de-ventas/

Saltos Intriago, D. A., & Villacis Breilh, O. J. (2022). *Implementación de Machine Learning en el área de ventas de la empresa Zapec S.A.* [Universidad Católica de Santiago de Guayaquil]. http://repositorio.ucsg.edu.ec/handle/3317/18337

Santiago, B. A., Macedo, P. B., & Andrade-Arenas, L. (2023). Implementation of a Web System with Machine Learning for Sentiment Analysis in Social Networks for the Marketing. *International Journal of Engineering Trends and Technology*, *71*(11), 45–55. https://doi.org/10.14445/22315381/IJETT-V71I11P205

Saravanan, N. P., Muthupriyanka, R., Kishore, K. C. N., & Ishwarya, S. (2023). Sales Prediction for Food Products Using Machine Learning. *2023 International Conference on Computer Communication and Informatics, ICCCI 2023*. https://doi.org/10.1109/ICCCI56745.2023.10128282

Schmidt, A., Kabir, M. W. U., & Hoque, M. T. (2022). Machine Learning Based Restaurant Sales Forecasting. In *Machine Learning and Knowledge Extraction* (Vol. 4, Issue 1, pp. 105–130). Multidisciplinary Digital Publishing Institute (MDPI). https://doi.org/10.3390/make4010006

Serrano Silvestre, P. (2021). *La Digitalización de las Compañías: Una Necesidad Para Sobrevivir*.

SitioBigData. (2018). *Aprendizaje Automático ml: Métricas de regresión*. SitioBigData. https://sitiobigdata.com/2018/08/27/machine-learning-metricas-regresion-mse/

StarTechUP. (2021, December 7). *10 Importantes Tecnologías de Desarrollo de Software que debe conocer*. StarTechUP. https://www.startechup.com/es/blog/tecnologias-de-desarrollo-de-software/#What_is_a_Software_Development_Technology

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction second edition* (2nd ed.).

Toro Ocampo, E. M., Mejía Giraldo, D. A., & Salazar Isaza, H. (2004). Pronóstico de Ventas Usando Redes Neuronales. *Scientia Et Technica*, *X*, 25–30.

UTEC. (2021, July 1). *La digitalización laboral tras la pandemia: El caso peruano*. UTEC. https://utec.edu.pe/blog-de-carreras/administracion-y-negocios-digitales/utec-peru-digitalizacion-laboral-tras-la-pandemia-coronavirus-herramientas-digitales-tecnologia

Valverde Bourdié, S. (2019). *Aplicaciones de la Inteligencia Artificial en la Empresa* [Universidad de Cantabria].

https://repositorio.unican.es/xmlui/bitstream/handle/10902/17521/VALVERDEBOURDIESANDRA.pdf

Villaseñor, C. (2024). *IA y automatización, el reto de la industria gastronómica*. CIO. https://iworld.com.mx/ia-y-automatizacion-el-reto-de-la-industria-gastronomica/

Vivas, H. (2014). *Optimización en el entrenamiento del Perceptrón Multicapa*. http://repositorio.unicauca.edu.co:8080/xmlui/handle/123456789/582

Yi, S. (2023). Walmart Sales Prediction Based on Machine Learning. *Highlights in Science, Engineering and Technology*, *47*, 87–94. https://doi.org/10.54097/hset.v47i.8170

ANEXOS

ANEXO 1. Código Fuente de los modelos de predicción.

Nombre de archivo: XGBoost.py

```
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
import xgboost as xgb
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score,
mean_absolute_percentage_error
from sklearn.preprocessing import StandardScaler
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from scipy import stats

#Cargar datos
df =
pd.read_csv('C:/Users/Rodrigo/Desktop/SISTEMA_CEVICHERIA/ventasCev_modificado.csv',
delimiter=',')
variable= 'producto_17'

#Calcular Z-scores
z_scores = stats.zscore(df[variable])

#Eliminar outliers
df= df[(z_scores >= -3) & (z_scores <= 3)]

#Convertir la columna Fecha a datetime
df['Fecha'] = pd.to_datetime(df['Fecha'])

#Características adicionales
df['Dia_de_semana'] = df['Fecha'].dt.dayofweek
df['Dia_del_mes'] = df['Fecha'].dt.day
df['Dia_del_año'] = df['Fecha'].dt.dayofyear
df['Año'] = df['Fecha'].dt.year
df['Mes'] = df['Fecha'].dt.month
df['Es_fin_de_semana'] = df['Dia_de_semana'].isin([5, 6]).astype(int)
df['Semana_del_año'] = df['Fecha'].dt.isocalendar().week
df['Trimestre'] = df['Fecha'].dt.quarter

df['Lag_1'] = df[variable].shift(1)
df['Lag_7'] = df[variable].shift(7)

#Medias móviles
df['Media_movil_3'] = df[variable].rolling(window=3).mean()
```

```
df['Media_movil_7'] = df[variable].rolling(window=7).mean()
df['Media_movil_30'] = df[variable].rolling(window=30).mean()
df['Diferencia_dia_anterior'] = df[variable].diff()

#df.dropna(inplace=True)  #Eliminar filas con valores NaN

plt.figure(figsize=(10, 6))
sns.histplot(df[variable], bins=30, kde=True)
plt.title('Distribución de Ventas de '+ variable)
plt.xlabel('Ventas')
plt.ylabel('Frecuencia')
plt.show()

print(df[[variable]].describe())

#Variables predictoras
X = df[['Dia_del_año', 'Año', 'Mes', 'Dia_de_semana', 'Es_fin_de_semana',
'Dia_del_mes', 'Semana_del_año', 'Trimestre', 'Lag_1', 'Lag_7', 'Media_movil_3',
'Media_movil_7', 'Media_movil_30', 'Diferencia_dia_anterior']]

#Variable objetivo
y = df[variable]

#División en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

#Crear y entrenar el modelo para la predicción de "y"

if variable != 'Total':
    model= xgb.XGBRegressor(
        objective='reg:squarederror',
        alpha= 0,
        gamma= 0.5,
        colsample_bytree= 1,
        subsample= 1,
        learning_rate= 0.3,
        n_estimators=300,
        seed=42,
        max_depth= 8)

else:
    model= xgb.XGBRegressor(
        objective='reg:squarederror',
        alpha= 0,
        colsample_bytree= 1.0,
        subsample= 0.8,
        learning_rate= 0.01,
        n_estimators=300,
```

```
        seed=42,
        max_depth= 6)

model.fit(X_train, y_train)

#Predicciones
y_pred = model.predict(X_test)

#r2
r2= r2_score(y_test, y_pred)

#MAE
mae = mean_absolute_error(y_test, y_pred)

# Calcular el error cuadrático medio (MSE)
mse= mean_squared_error(y_test, y_pred)

# MSE en el conjunto de entrenamiento
y_train_pred = model.predict(X_train)
mse_train = mean_squared_error(y_train, y_train_pred)

# MSE en el conjunto de prueba
y_test_pred = model.predict(X_test)
mse_test = mean_squared_error(y_test, y_test_pred)

#Validación cruzada con el modelo ajustado
cv_scores = cross_val_score(model, X, y, cv=5, scoring='neg_mean_squared_error')
cv_mse = -cv_scores.mean()

#MAPE
epsilon = 1e-10
mape = mean_absolute_percentage_error(y_test + epsilon, y_test_pred + epsilon)

#SMAPE
def smape(y_true, y_pred):
    return 100 * np.mean(2 * np.abs(y_pred - y_true) / (np.abs(y_pred) +
np.abs(y_true)))

smape_value = smape(y_test, y_test_pred)

print(f'MAE en el conjunto de prueba: {mae}')
print(f'MSE en el conjunto de entrenamiento: {mse_train}')
print(f'MSE en el conjunto de prueba: {mse_test}')
print(f'MSE con validación cruzada: {cv_mse}')
print("R2 de ", variable, ": ", r2)
print(f'MAPE: {mape}')
print(f'SMAPE: {smape_value}')

#Análisis Residual
```

```
residuals = y_test - y_test_pred
plt.figure(figsize=(10, 6))
sns.histplot(residuals, kde=True)
plt.title('Distribución de los Residuos')
plt.xlabel('Residuos')
plt.show()

# Gráfico de Residuos vs. Valores Predichos
plt.figure(figsize=(10, 6))
plt.scatter(y_test_pred, residuals, alpha=0.5)
plt.axhline(0, color='r', linestyle='--')
plt.title('Residuos vs. Valores Predichos')
plt.xlabel('Valores Predichos')
plt.ylabel('Residuos')
plt.show()

# Visualización de Predicciones vs Valores Reales
plt.figure(figsize=(14, 8))
plt.plot(y_test.values, label='Valores Reales', color='b')
plt.plot(y_test_pred, label='Predicciones', color='r')
plt.xlabel('Índice')
plt.ylabel('Cantidad')
plt.title('Valores Reales vs Predicciones')
plt.legend()
plt.show()

#Importancia de características
plt.figure(figsize=(16, 8))
xgb.plot_importance(model, max_num_features=10)
plt.title('Importancia de las Características')
plt.show()

#Fechas futuras
dias_futuros = pd.date_range(start='2024-02-29', periods=60)
futuro_df = pd.DataFrame(dias_futuros, columns=['Fecha'])

#Características para fechas futuras
futuro_df['Dia_del_año'] = futuro_df['Fecha'].dt.dayofyear
futuro_df['Año'] = futuro_df['Fecha'].dt.year
futuro_df['Mes'] = futuro_df['Fecha'].dt.month
futuro_df['Dia_de_semana'] = futuro_df['Fecha'].dt.dayofweek
futuro_df['Es_fin_de_semana'] = futuro_df['Dia_de_semana'].isin([5, 6]).astype(int)

futuro_df['Dia_del_mes'] = df['Fecha'].dt.day
futuro_df['Semana_del_año'] = df['Fecha'].dt.isocalendar().week
futuro_df['Trimestre'] = df['Fecha'].dt.quarter

futuro_df['Media_movil_3'] = df[variable].rolling(window=3).mean()
futuro_df['Media_movil_7'] = df[variable].rolling(window=7).mean()
```

```
futuro_df['Media_movil_30'] = df[variable].rolling(window=30).mean()
futuro_df['Diferencia_dia_anterior'] = df[variable].diff()

last_known_values = df[['Fecha', variable]].sort_values(by='Fecha').tail(7)
futuro_df = futuro_df.merge(last_known_values, how='left', on='Fecha')

futuro_df['Lag_1'] = futuro_df[variable].shift(1)
futuro_df['Lag_7'] = futuro_df[variable].shift(7)
futuro_df.fillna(0, inplace=True)  # Rellenar NaNs con 0

#Hacer predicciones
X_futuro = futuro_df[['Dia_del_año', 'Año', 'Mes', 'Dia_de_semana',
'Es_fin_de_semana', 'Dia_del_mes', 'Semana_del_año', 'Trimestre', 'Lag_1', 'Lag_7',
'Media_movil_3', 'Media_movil_7', 'Media_movil_30', 'Diferencia_dia_anterior']]
pred_futuro = model.predict(X_futuro)

#Agregar predicciones al DataFrame
futuro_df[variable + '_Pred'] = pred_futuro

print(futuro_df)

sns.set(style="whitegrid")

#Crear el gráfico
plt.figure(figsize=(14, 8))

#Graficar las predicciones
plt.plot(futuro_df['Fecha'], futuro_df[variable + '_Pred'], marker='o', linestyle='-
', color='b', label='Predicción '+ variable)

#Configurar el gráfico
plt.xlabel('Fecha')
plt.ylabel('Predicciones')
plt.title('Predicciones de Ventas Futuras')
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()

#Mostrar el gráfico
plt.show()
```

Nombre de archivo: SARIMA.py

```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
```

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score,
mean_absolute_percentage_error
from scipy import stats
import numpy as np

# Cargar datos y convertir 'Fecha' en índice
df =
pd.read_csv('C:/Users/Rodrigo/Desktop/SISTEMA_CEVICHERIA/ventasCev_modificado.csv',
delimiter=',')
df['Fecha'] = pd.to_datetime(df['Fecha'], format='%Y-%m-%d')
df.set_index('Fecha', inplace=True)
print(df.head())

variable= "producto_18"

#Calcular Z-scores
z_scores = stats.zscore(df[variable])
outliers_z = df[(z_scores < -3) | (z_scores > 3)]

#Eliminar outliers
df= df[(z_scores >= -3) & (z_scores <= 3)]


#Características temporales adicionales
df['Dia_de_semana'] = df.index.dayofweek
df['Dia_del_mes'] = df.index.day
df['Dia_del_año'] = df.index.dayofyear
df['Año'] = df.index.year
df['Mes'] = df.index.month
df['Es_fin_de_semana'] = df['Dia_de_semana'].isin([5, 6]).astype(int)
df['Semana_del_año'] = df.index.isocalendar().week
df['Trimestre'] = df.index.quarter

#Lag Variables
df['Lag_1'] = df[variable].shift(1)
df['Lag_7'] = df[variable].shift(7)

#Medias móviles
df['Media_movil_3'] = df[variable].rolling(window=3).mean()
df['Media_movil_7'] = df[variable].rolling(window=7).mean()
df['Media_movil_30'] = df[variable].rolling(window=30).mean()
df['Diferencia_dia_anterior'] = df[variable].diff()

#Eliminar filas con valores nulos
df.dropna(inplace=True)

X = df[[variable, 'Dia_de_semana', 'Dia_del_mes', 'Dia_del_año', 'Año', 'Mes',
            'Es_fin_de_semana', 'Semana_del_año', 'Trimestre', 'Lag_1', 'Lag_7',
```

```
            'Media_movil_3', 'Media_movil_7', 'Media_movil_30',
'Diferencia_dia_anterior']].resample('D').sum()

print(X.head())

#GRAFICAR MESES
plt.figure(figsize=(12, 6))
plt.plot(X[variable], linewidth=1,c='blue')
plt.title("Venta")
plt.xlabel("Fecha")
plt.ylabel(variable)
plt.show()

def estacionariedad(timeseries):
    #Dickey-Fuller test
    result = adfuller(timeseries, autolag='AIC')
    p_value = result[1] #El p-valor obtenido rechaza la H0 y concluye que la serie es
estacionaria, por ende d = 0
    print(f'ADF Statistic: {result[0]}')
    print(f'p-value: {p_value}')
    print('Estacionaria' if p_value < 0.05 else 'No Estacionaria')


estacionariedad(X[variable])

#ACF y PACF para la serie temporal
plot_acf(X[variable])
plot_pacf(X[variable])
plt.show()

train_size = int(len(X) * 0.8)  #80% para entrenamiento
train, test = X[0:train_size], X[train_size:len(X)]  #Dividir los datos

#Ajustar el modelo en el conjunto de entrenamiento
p, d, q = 1, 0, 0  # Parámetros del modelo
P, D, Q, s = 1, 0, 0, 7  # Parámetros estacionales

#Definir las características exógenas para el conjunto de entrenamiento
exogenous_train = train[['Dia_de_semana', 'Mes', 'Es_fin_de_semana', 'Lag_1',
'Media_movil_7']]

#Entrenar modelo
model = SARIMAX(train[variable],
                order=(p, d, q),
                seasonal_order=(P, D, Q, s),
                exog=exogenous_train)
results = model.fit()

print(results.summary())
```

```
#Pronosticar sobre el conjunto de prueba
periodos_pronostico= len(test)
pronostico_exog= test[['Dia_de_semana', 'Mes', 'Es_fin_de_semana', 'Lag_1',
'Media_movil_7']]
pronostico= results.get_forecast(steps=periodos_pronostico, exog=pronostico_exog)
media_pronostico= pronostico.predicted_mean
pronostico_ci= pronostico.conf_int()

plt.figure(figsize=(12, 6))
plt.plot(train[variable], label='Entrenamiento', color='blue')
plt.plot(test[variable], label='Prueba', color='green')
plt.plot(media_pronostico, label='Pronóstico', color='red')
plt.fill_between(pronostico_ci.index, pronostico_ci.iloc[:, 0], pronostico_ci.iloc[:,
1], color='pink')
plt.title("Pronóstico de ventas de " + variable)
plt.xlabel("Fecha")
plt.ylabel(variable)
plt.legend()
plt.show()

print(df[[variable]].describe())

mae = mean_absolute_error(test[variable], media_pronostico)
mse = mean_squared_error(test[variable], media_pronostico)
r2 = r2_score(test[variable], media_pronostico)
def modified_mape(y_true, y_pred):
    mask = y_true != 0
    return np.mean(np.abs((y_true[mask] - y_pred[mask]) / y_true[mask])) * 100

mape = modified_mape(test[variable], media_pronostico)

print(f'MAE: {mae}')
print(f'MSE: {mse}')
print(f'R²: {r2}')
print(f'MAPE: {mape:.2f}%')
```

Nombre de archivo: LSTM.py

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import r2_score, mean_absolute_error
import matplotlib.pylab as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.callbacks import EarlyStopping
```

```
from scipy import stats


df=
pd.read_csv('C:/Users/Rodrigo/Desktop/SISTEMA_CEVICHERIA/ventasCev_modificado.csv',
delimiter=',')
print(df.head())

#Convertir la columna Fecha a datetime
df['Fecha'] = pd.to_datetime(df['Fecha'])
df.set_index('Fecha', inplace=True)

variable= 'Total'

#Calcular Z-scores
z_scores = stats.zscore(df[variable])

#Eliminar outliers
df= df[(z_scores >= -3) & (z_scores <= 3)]

series= df[variable]

#Escalar
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_series = scaler.fit_transform(series.values.reshape(-1, 1))

#Crear una secuencia de datos para LSTM
def create_dataset(data, time_step=1):
    X, Y = [], []
    for i in range(len(data) - time_step - 1):
        a = data[i:(i + time_step), 0]
        X.append(a)
        Y.append(data[i + time_step, 0])
    return np.array(X), np.array(Y)

#Definir el tamaño de los pasos de tiempo
time_step = 10
X, Y = create_dataset(scaled_series, time_step)

#Dividir los datos en entrenamiento y prueba
train_size = int(len(X) * 0.8)
test_size = len(X) - train_size
X_train, X_test = X[0:train_size], X[train_size:len(X)]
Y_train, Y_test = Y[0:train_size], Y[train_size:len(Y)]

#Remodelar los datos
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)
```

```
#Crear el modelo LSTM
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(time_step, 1)))
model.add(LSTM(50, return_sequences=False))
model.add(Dense(1))

#Compilar el modelo
model.compile(optimizer='adam', loss='mean_absolute_error')

#Entrenar el modelo
early_stopping = EarlyStopping(monitor='val_loss', patience=10)
model.fit(X_train, Y_train, epochs=100, batch_size=32, validation_data=(X_test,
Y_test), callbacks=[early_stopping], verbose=1)

#Hacer predicciones
train_predict = model.predict(X_train)
test_predict = model.predict(X_test)

#Desescalar las predicciones
train_predict = scaler.inverse_transform(train_predict)
test_predict = scaler.inverse_transform(test_predict)

#Desescalar los datos reales
Y_train = scaler.inverse_transform([Y_train])
Y_test = scaler.inverse_transform([Y_test])

#MSE
test_mse = np.mean((Y_test[0] - test_predict[:, 0])**2)

#R2
test_r2 = r2_score(Y_test[0], test_predict[:, 0])

#MAE
test_mae = mean_absolute_error(Y_test[0], test_predict[:, 0])

#MAPE
def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

test_mape = mean_absolute_percentage_error(Y_test[0], test_predict[:, 0])

#SMAPE
def symmetric_mean_absolute_percentage_error(y_true, y_pred):
    return np.mean(2 * np.abs(y_pred - y_true) / (np.abs(y_true) + np.abs(y_pred))) *
100

test_smape = symmetric_mean_absolute_percentage_error(Y_test[0], test_predict[:, 0])
```

```
#Mostrar los resultados
print(f'MAE en el conjunto de prueba: {test_mae}')
print(f'MSE en el conjunto de prueba: {test_mse}')
print(f'R² en el conjunto de prueba: {test_r2}')
print(f'MAPE en el conjunto de prueba: {test_mape}')
print(f'SMAPE en el conjunto de prueba: {test_smape}')

#Graficar los resultados
plt.figure(figsize=(12, 6))
plt.plot(series.index, series, label='Datos Reales')
train_predict_plot = np.empty_like(series)
train_predict_plot[:] = np.nan
train_predict_plot[time_step:len(train_predict) + time_step] = train_predict[:, 0]

test_predict_plot = np.empty_like(series)
test_predict_plot[:] = np.nan

#Calcular los índices de inicio y fin para test_predict_plot
test_start = len(train_predict) + (time_step * 2)
test_end = test_start + len(test_predict)

#Ajustar los índices
if test_end > len(test_predict_plot):
    test_end = len(test_predict_plot)

test_predict_plot[test_start:test_end] = test_predict[:test_end - test_start, 0]

plt.plot(series.index, train_predict_plot, label='Predicciones de Entrenamiento')
plt.plot(series.index, test_predict_plot, label='Predicciones de Prueba')
plt.legend()
plt.show()

#Predecir los próximos 30 días
x_input = test_predict[-time_step:].reshape(1, -1)
temp_input = list(x_input)
temp_input = temp_input[0].tolist()

lst_output = []
n_steps = time_step
i = 0
while(i < 30):
    if(len(temp_input) > time_step):
        x_input = np.array(temp_input[1:])
        x_input = x_input.reshape((1, time_step, 1))
        yhat = model.predict(x_input, verbose=0)
        temp_input.extend(yhat[0].tolist())
        temp_input = temp_input[1:]
        lst_output.extend(yhat.tolist())
        i = i + 1
```

```
    else:
        x_input = x_input.reshape((1, time_step, 1))
        yhat = model.predict(x_input, verbose=0)
        temp_input.extend(yhat[0].tolist())
        lst_output.extend(yhat.tolist())
        i = i + 1

#Desescalar las predicciones futuras
future_predictions = scaler.inverse_transform(lst_output)

#Crear un DataFrame para las predicciones futuras
future_dates = pd.date_range(start=series.index[-1] + pd.Timedelta(days=1),
periods=30, freq='D')
future_df = pd.DataFrame(future_predictions, index=future_dates,
columns=['Predicción'])

#Graficar las predicciones futuras
plt.figure(figsize=(12, 6))
plt.plot(series.index, series, label='Datos Históricos')
plt.plot(future_df, label='Predicciones Futuras', linestyle='dashed')
plt.legend()
plt.show()
```

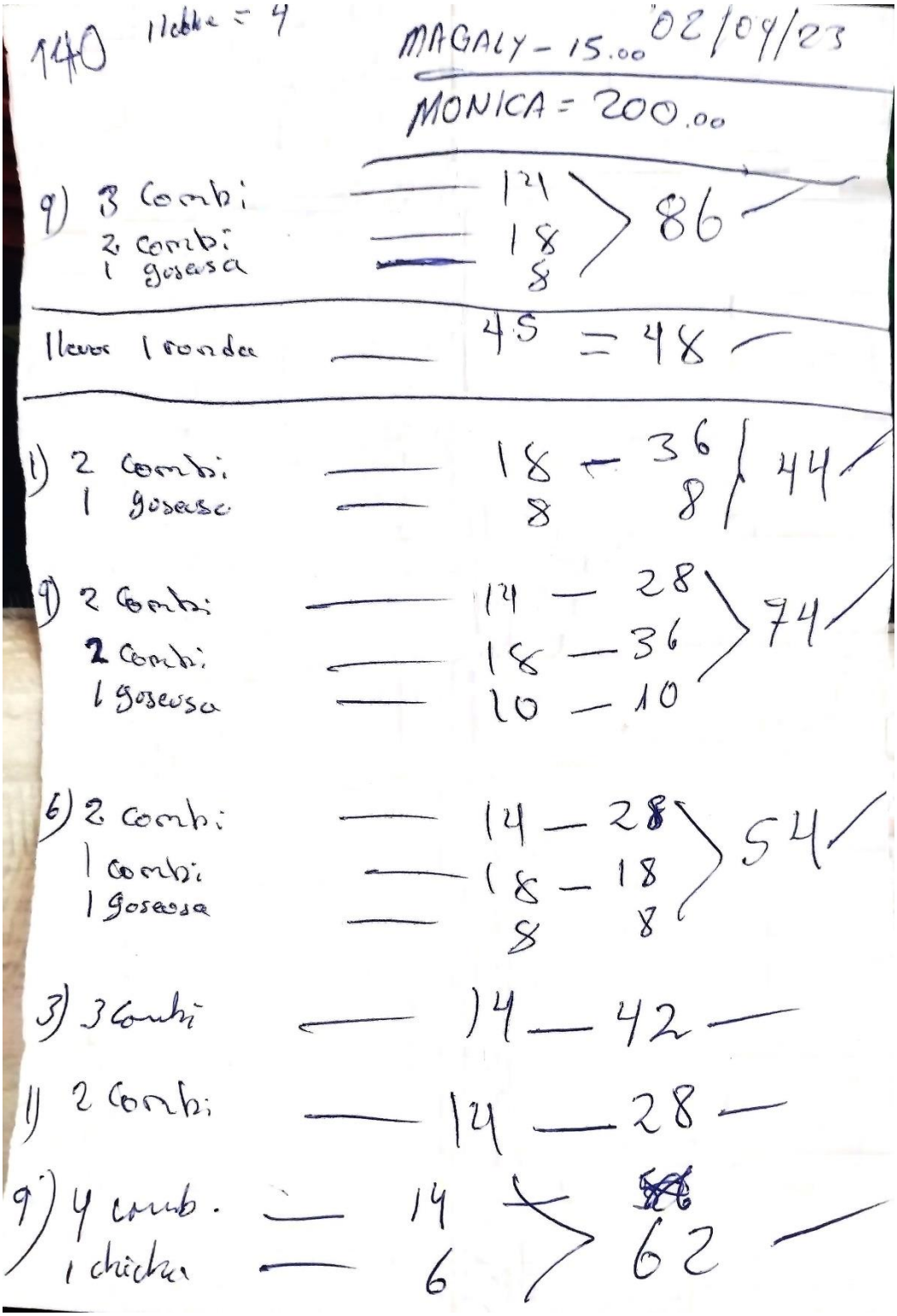
Nombre de archivo: MLP.py

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error,
mean_absolute_percentage_error
from sklearn.linear_model import Lasso
from sklearn.pipeline import Pipeline
import matplotlib.pyplot as plt
from scipy import stats

#Cargar los datos
df =
pd.read_csv('C:/Users/Rodrigo/Desktop/SISTEMA_CEVICHERIA/ventasCev_modificado.csv',
delimiter=',')

#Convertir la columna 'Fecha' a datetime y establecerla como índice
df['Fecha'] = pd.to_datetime(df['Fecha'])
df.set_index('Fecha', inplace=True)

variable= 'Total'

#Calcular Z-scores
z_scores = stats.zscore(df[variable])
```

```
#Eliminar outliers
df= df[(z_scores >= -3) & (z_scores <= 3)]

series = df[variable]

print(series.describe())

#Escalar los datos entre 0 y 1
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_series = scaler.fit_transform(series.values.reshape(-1, 1))

#Función para crear una secuencia de datos para MLP
def create_dataset(data, time_step=1):
    X, Y = [], []
    for i in range(len(data) - time_step - 1):
        a = data[i:(i + time_step), 0]
        X.append(a)
        Y.append(data[i + time_step, 0])
    return np.array(X), np.array(Y)

#Definir el tamaño de los pasos de tiempo
time_step = 10
X, Y = create_dataset(scaled_series, time_step)

#Dividir los datos en entrenamiento y prueba
train_size = int(len(X) * 0.8)
test_size = len(X) - train_size
X_train, X_test = X[0:train_size], X[train_size:len(X)]
Y_train, Y_test = Y[0:train_size], Y[train_size:len(Y)]

#Crear el modelo MLP
mlp = MLPRegressor(hidden_layer_sizes=(100, 100), alpha=0.0001,
learning_rate_init=0.001, activation='relu', max_iter=2000, random_state=42)

#Entrenar el modelo
mlp.fit(X_train, Y_train)

#Hacer predicciones
train_predict = mlp.predict(X_train)
test_predict = mlp.predict(X_test)

#Desescalar las predicciones
train_predict = scaler.inverse_transform(train_predict.reshape(-1, 1))
test_predict = scaler.inverse_transform(test_predict.reshape(-1, 1))

#Desescalar los datos reales
Y_train = scaler.inverse_transform(Y_train.reshape(-1, 1))
Y_test = scaler.inverse_transform(Y_test.reshape(-1, 1))
```

```
#Calcular el MSE MAE y MAPE en el conjunto de prueba
test_mse = mean_squared_error(Y_test, test_predict)
test_mae = mean_absolute_error(Y_test, test_predict)

#Calcular el R2 en el conjunto de prueba
r2_test = r2_score(Y_test, test_predict)

#MAPE
def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

test_mape = mean_absolute_percentage_error(Y_test[0], test_predict[:, 0])

#SMAPE
def symmetric_mean_absolute_percentage_error(y_true, y_pred):
    return np.mean(2 * np.abs(y_pred - y_true) / (np.abs(y_true) + np.abs(y_pred))) *
100

test_smape = symmetric_mean_absolute_percentage_error(Y_test[0], test_predict[:, 0])

#Imprimir las métricas
print(f'MAE en el conjunto de prueba: {test_mae}')
print(f'MSE en el conjunto de prueba: {test_mse}')
print(f'R2 en el conjunto de prueba: {r2_test}')
print(f'MAPE en el conjunto de prueba: {test_mape}')
print(f'SMAPE en el conjunto de prueba: {test_smape}')
```

ANEXO 2. Apuntes realizados a mano de la cevichería Rico Norte.



ANEXO 3. Encuesta de Requerimientos y Necesidades Para la Implementación de un Sistema Informático Inteligente para la Cevichería Rico Norte.

ENCUESTA DE REQUERIMIENTOS Y NECESIDADES PARA LA IMPLEMENTACIÓN DE UN SISTEMA INFORMÁTICO INTELIGENTE PARA LA CEVICHERÍA RICO NORTE

1. ¿Cuáles son los principales objetivos de la Cevichería Rico Norte a corto y largo plazo?

2. ¿Qué aspectos del negocio le gustaría mejorar con la implementación de un sistema informático?

3. ¿Cuáles son los principales desafíos que enfrenta actualmente en la gestión del negocio?

4. ¿Qué procesos le resultan más lentos o problemáticos en la operación diaria del negocio?

5. ¿Qué tipo de datos recopila actualmente sobre las ventas u operaciones en general?

6. ¿Cómo se registran y almacenan estos datos?

7. ¿Qué espera obtener al implementar un sistema informático inteligente?

8. ¿Cuáles son las funcionalidades más importantes que le gustaría que tenga el sistema?

9. ¿Qué resultados específicos le gustaría ver después de la implementación del sistema?

10. ¿Tiene planes de expandir el negocio en el futuro?

ANEXO 4. Declaración jurada de autenticidad de tesis.

Universidad Nacional del Altiplano Puno | Vicerrectorado de Investigación | Repositorio Institucional

DECLARACIÓN JURADA DE AUTENTICIDAD DE TESIS

Por el presente documento, Yo Rodrigo Alexander Becerra Lucano,
identificado con DNI 74761925 en mi condición de egresado de:

☒ **Escuela Profesional,** ☐ **Programa de Segunda Especialidad,** ☐ **Programa de Maestría o Doctorado**

de Ingeniería Estadística e Informática,

informo que he elaborado el/la ☒ **Tesis** o ☐ **Trabajo de Investigación** denominada:
"Implementación de un Sistema Informático Inteligente para el Pronóstico de Ventas de la Cevichería Rico Norte de la Ciudad de Puno 2024"

Es un tema original.

Declaro que el presente trabajo de tesis es elaborado por mi persona y **no existe plagio/copia** de ninguna naturaleza, en especial de otro documento de investigación (tesis, revista, texto, congreso, o similar) presentado por persona natural o jurídica alguna ante instituciones académicas, profesionales, de investigación o similares, en el país o en el extranjero.

Dejo constancia que las citas de otros autores han sido debidamente identificadas en el trabajo de investigación, por lo que no asumiré como suyas las opiniones vertidas por terceros, ya sea de fuentes encontradas en medios escritos, digitales o Internet.

Asimismo, ratifico que soy plenamente consciente de todo el contenido de la tesis y asumo la responsabilidad de cualquier error u omisión en el documento, así como de las connotaciones éticas y legales involucradas.

En caso de incumplimiento de esta declaración, me someto a las disposiciones legales vigentes y a las sanciones correspondientes de igual forma me someto a las sanciones establecidas en las Directivas y otras normas internas, así como las que me alcancen del Código Civil y Normas Legales conexas por el incumplimiento del presente compromiso

Puno 25 de setiembre del 2024

FIRMA (obligatoria)

Huella

ANEXO 5. Autorización para el depósito de tesis en el Repositorio Institucional.

Universidad Nacional del Altiplano Puno | VRI Vicerrectorado de Investigación | Repositorio Institucional

AUTORIZACIÓN PARA EL DEPÓSITO DE TESIS O TRABAJO DE INVESTIGACIÓN EN EL REPOSITORIO INSTITUCIONAL

Por el presente documento, Yo Rodrigo Alexander Becerra Lucano,
identificado con DNI 74761925 en mi condición de egresado de:

☒ **Escuela Profesional,** ☐ **Programa de Segunda Especialidad,** ☐ **Programa de Maestría o Doctorado**

de Ingeniería Estadística e Informática,
informo que he elaborado el/la ☒ **Tesis** o ☐ **Trabajo de Investigación** denominada:

"Implementación de un Sistema Informático Inteligente para el pronóstico de Ventas de la Cevichería Rico Norte de la Ciudad de Puno 2024"

para la obtención de ☐**Grado,** ☒ **Título Profesional** o ☐ **Segunda Especialidad.**

Por medio del presente documento, afirmo y garantizo ser el legítimo, único y exclusivo titular de todos los derechos de propiedad intelectual sobre los documentos arriba mencionados, las obras, los contenidos, los productos y/o las creaciones en general (en adelante, los "Contenidos") que serán incluidos en el repositorio institucional de la Universidad Nacional del Altiplano de Puno.

También, doy seguridad de que los contenidos entregados se encuentran libres de toda contraseña, restricción o medida tecnológica de protección, con la finalidad de permitir que se puedan leer, descargar, reproducir, distribuir, imprimir, buscar y enlazar los textos completos, sin limitación alguna.

Autorizo a la Universidad Nacional del Altiplano de Puno a publicar los Contenidos en el Repositorio Institucional y, en consecuencia, en el Repositorio Nacional Digital de Ciencia, Tecnología e Innovación de Acceso Abierto, sobre la base de lo establecido en la Ley N° 30035, sus normas reglamentarias, modificatorias, sustitutorias y conexas, y de acuerdo con las políticas de acceso abierto que la Universidad aplique en relación con sus Repositorios Institucionales. Autorizo expresamente toda consulta y uso de los Contenidos, por parte de cualquier persona, por el tiempo de duración de los derechos patrimoniales de autor y derechos conexos, a título gratuito y a nivel mundial.

En consecuencia, la Universidad tendrá la posibilidad de divulgar y difundir los Contenidos, de manera total o parcial, sin limitación alguna y sin derecho a pago de contraprestación, remuneración ni regalía alguna a favor mío; en los medios, canales y plataformas que la Universidad y/o el Estado de la República del Perú determinen, a nivel mundial, sin restricción geográfica alguna y de manera indefinida, pudiendo crear y/o extraer los metadatos sobre los Contenidos, e incluir los Contenidos en los índices y buscadores que estimen necesarios para promover su difusión.

Autorizo que los Contenidos sean puestos a disposición del público a través de la siguiente licencia:

Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional. Para ver una copia de esta licencia, visita: https://creativecommons.org/licenses/by-nc-sa/4.0/

En señal de conformidad, suscribo el presente documento.

Puno 25 de setiembre del 2024

FIRMA (obligatoria)

Huella