UNIVERSIDAD NACIONAL DEL ALTIPLANO

FACULTAD DE INGENIERÍA MECÁNICA ELÉCTRICA, ELECTRÓNICA Y SISTEMAS

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

MODELO BASADO EN MACHINE LEARNING PARA OPTIMIZAR EL PRONÓSTICO DE VENTAS DE LA EMPRESA RICOS PAN, AÑO 2020 – 2021

TESIS

PRESENTADA POR:

GERSON MANUEL PONCE ILLACUTIPA

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO DE SISTEMAS

PUNO – PERÚ

2022

Reporte de similitud

NOMBRE DEL TRABAJO

MODELO BASADO EN MACHINE LEARNING PARA OPTIMIZAR EL PRONÓSTICO DE VENTAS DE LA EMPRESA RICOS PAN , AÑO 2020 - 2021

AUTOR

GERSON MANUEL PONCE ILLACUTIPA

RECUENTO DE PALABRAS

20936 Words

RECUENTO DE CARACTERES

116976 Characters

RECUENTO DE PÁGINAS

107 Pages

TAMAÑO DEL ARCHIVO

2.3MB

FECHA DE ENTREGA

Nov 7, 2022 11:58 AM GMT-5

FECHA DEL INFORME

Nov 7, 2022 11:59 AM GMT-5

10% de similitud general

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

- 9% Base de datos de Internet
- 2% Base de datos de publicaciones
- Base de datos de Crossref
- Base de datos de contenido publicado de Crossr
- 7% Base de datos de trabajos entregados

Excluir del Reporte de Similitud

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

Universidad Nacional del Altiplano

Firmado digitalmente por HOLGUIN HOLGUIN Edgar FAU 20145496170 hard
Motivo: Soy el autor del documento
Fecha: 12.09.2023 10:07:36 -05:00

Resumen

DEDICATORIA

A mis padres Manuel Ponce y Ofelia Illacutipa quienes por su esfuerzo y sacrificio siempre me apoyaron en cada paso de mis estudios, brindándome así la oportunidad de ser un profesional, a mis hermanas Maribi y Lizeth, quienes durante este camino me brindaron su guía y apoyo.

AGRADECIMIENTOS

Agradezco a la Universidad Nacional del Altiplano, en especial a la Escuela Profesional de Ingeniería de Sistemas quien nos prepara para ser grandes profesionales y competitivos para la servir a nuestra sociedad.

A los docentes de la Escuela Profesional de Ingeniería de sistemas, quienes siempre nos impartieron los conocimientos y experiencias.

A mi Asesor y director de tesis M.Sc. Edgar Holguin Holguin, por su orientación y apoyo constante durante el desarrollo de la presente investigación.

Mi más sincero agradecimiento a los jurados de mi proyecto de investigación, a M.Sc. William Eusebio Arcaya Coaquira, a M.Sc. Magali Gianina Gonzales Paco, a M.Sc. Fidel Huanco Ramos por las sugerencias y correcciones.

Así mismo a la Entidad Privada Ricos Pan Puno, quienes son ente principal para este objeto de estudio y que me permitieron el acopio de la información necesaria para llevar adelante mi trabajo de investigación.

ÍNDICE GENERAL

Área : Ingeniería de software, bases de datos e inteligencia de negocios

Línea : Machine Learning

Fecha de sustentación: 10 de noviembre de 2022

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ACRÓNIMOS

ARIMA	Modelo autorregresivo integrado de promedio móvil (Autoregressive integrated moving average)
CNN	Redes neuronales convolucionales (Convolutional neural network)
CSV	Valores separados por comas (comma-separated values)
GRU	Unidad Recurrente Cerrada (Gated Recurrent Unit)
JSON	Notación de objeto de JavaScript (JavaScript Object Notation)
LSTM	Memoria a corto y largo plazo (Long-Short Term Memory)
RNN	Red neuronal recurrente (Recurrent Neural Network)
SVM	Máquina de soporte vectorial (Support Vector Machines)
SVR	Maquina de soporte vectorial para regresión (Support Vector Regression)

RESUMEN

En un mundo competitivo la capacidad de una empresa para alinearse a los cambios del mercado es un factor decisivo, en una empresa donde las ventas son de vital importancia, el pronóstico de ellas juega un papel fundamental. En los últimos años existe un creciente interés en el Machine Learning, en esta investigación se presenta un modelo basado en Machine Learning que optimiza el pronóstico de ventas para la empresa Ricos Pan en la ciudad de Puno. El proceso de investigación se llevó a cabo en los años 2020 y 2021, para dicho objetivo se construyó el conjunto de datos a partir de los registros de venta que proporcionó la empresa, se realizó un preprocesamiento a los datos donde se observó registros faltantes debido a que la empresa cerro temporalmente por la pandemia Covid – 19, también se observaron ventas atípicas en los días festivos como el día de la madre, el día del padre, los fines de semana, para tratar este tipo de características se utilizaron diversos métodos como el diagrama de cajas y bigotes, rangos intercuantiles, adición de atributos entre otras técnicas de preprocesamiento, por otro lado se entrenaron y probaron modelos basados en redes neuronales recurrentes (LSTM y GRU), redes neuronales convolucionales (CNN) y un modelo basado en máquinas de vectores soporte (SVR), para la etapa de entrenamiento se probaron distintas épocas y distintos tipos de transformación de datos utilizando logaritmos y diferencia simple, llegando a la conclusión que el modelo basado en redes neuronales convolucionales tuvo mejores resultados a diferencia de otros modelos, así mismo se comparó el pronóstico manual que se realiza en la empresa con el modelo propuesto obteniendo un resultado muy superior, para medir el desempeño de cada modelo y el pronóstico manual se usó el error porcentual absoluto medio.

Palabras Clave: Series de tiempo, Pronostico venta, Machine Learning, redes neuronales.

ABSTRACT

In a competitive world the ability of a company to align to market changes is a decisive factor, in a company where sales are of vital importance, the forecast of them plays a key role in recent years there is a growing interest in Machine Learning, in this research a model based on Machine Learning that optimizes the sales forecast for the company Ricos Pan in the Puno city is presented, The research process was carried out in the years 2020 and 2021, for this objective the data set was built from the sales records provided by the company, a preprocessing was performed to the data where missing data was observed because the company temporarily closed due to the pandemic Covid - 19, also atypical sales were observed on holidays such as Mother's Day, Father's Day, weekends, to deal with this type of characteristics, several methods were used such as box plot, interquantile ranges, addition of attributes among other preprocessing techniques, on the other hand, models based on recurrent neural networks (LSTM and GRU), convolutional neural networks (CNN) and a model based on support vector machines (SVR) were entered and tested, For the training stage, different epochs and different types of data transformation were tested using logarithms and simple difference, reaching the conclusion that the model based on convolutional neural networks had better results than other models, likewise the manual forecast that is performed in the company was compared with the proposed model obtaining a much better result, to measure the performance of each model and the manual forecast the mean absolute percentage error was used.

Keywords: Time series, Sales Forecast, Machine Learning, neuronal network.

CAPÍTULO I

INTRODUCCIÓN

El pronóstico de sucesos ha sido un tema relevante desde tiempo remotos, para poder pronosticar datos se requiere una búsqueda de acciones pasadas. Los pronósticos son fundamentales para toda empresa pues constituyen el fundamento de su quehacer a mediano y a largo plazo, es fácil comprobar que las áreas funcionales de las empresas requieren a diario de los pronósticos para planear su actividad, es importante resaltar la tarea de realizar previsiones empresariales, entendida como una herramienta que permite pronosticar factores relevantes como el de las ventas (Cadena Lozano, 2016). Los modelos de Machine Learning se han establecido en la última como serios rivales a la estadística clásica en el campo de pronóstico de series de tiempo (Ahmed et al., 2010).

La empresa Ricos Pan como grupo gastronómico cuenta con una cadena de panaderías, cafeterías, viene también introduciéndose en la era de la digitalización; debido a que actualmente posee un sistema de información para el control de ventas e inventario, dicho sistema viene acumulando datos históricos, que no están siendo aprovechados para mejorar sus estrategias operativas, como el pronóstico de ventas, lo cual influye también en el volumen producción y el desperdicio de producto.

La presente investigación tiene como objetivo general optimizar el pronóstico de ventas a través de un modelo basado en Machine Learning en la empresa Ricos Pan la investigación se desarrolló con los datos de los años 2020 y 2021, para el desarrollo del modelo se tomó en cuenta máquina de soporte de vectores, redes neuronales recurrentes y redes neuronales convolucionales.

En el Capítulo I se plantea el problema para conocer los motivos que originaron esta investigación y los objetivos desarrollados, así como las limitaciones que se tuvieron.

En el Capítulo II se describen los antecedentes donde se muestran investigaciones que se hicieron respecto al pronóstico de ventas como también investigaciones que compararon diversos métodos que usan las empresas, para una mejor compresión del objeto de estudio se desarrolló el marco teórico en donde se abordara acerca de las técnicas de pronóstico se tomó con mucha énfasis las series de tiempo ya que es la técnica que se usara en la presente investigación, del mismo modo se abordó los métodos de Machine Learning y las métricas que se usan para poder medir, por último se desarrolló la hipótesis y se definieron las variables de la investigación.

En el Capítulo III se muestra los materiales y métodos, así como el tipo y diseño de la investigación; en este capítulo se muestra el desarrollo paso a paso de los objetivos específicos, así como el preprocesamiento de datos, la implementación de los modelos basado en Machine Learning, la forma de medición de los modelos y se abordara el pronóstico manual que se desarrolla en la empresa analizada.

En el Capítulo IV, V y VI se muestran los resultados a través de figuras y tablas, así como discusión, finalmente se hace las recomendaciones.

1.1. DESCRIPCIÓN DEL PROBLEMA

El pronóstico de ventas es de suma importancia porque es fuente información importante utilizada para toma de decisiones en cuanto a la operación, producción y planificación en diferentes industrias, el no contar con una herramienta para el pronóstico de ventas puede dar como resultado problemas de exceso o insuficiencia de inventario, desperdicio de productos, tiempo innecesario en la realización de los diferentes procesos (Cadena Lozano, 2016).

La empresa Ricos Pan viene también introduciéndose en la era de la digitalización; debido a que actualmente posee un sistema de información digital para el control de ventas e inventario, dicho sistema viene acumulando datos históricos, que no están siendo aprovechados para mejorar sus estrategias operativas. Actualmente el pronóstico de ventas se viene realizando de forma manual esto impacta directamente en la inversión de horas hombre y en la operación, el pronóstico se realiza solo con la información de días anteriores o el mes anterior, sin embargo no se toma en cuenta el historial de ventas, dando como resultado un pronóstico que no se ajusta a la realidad, lo cual genera inexactitudes en la producción y merma en los productos, además que se invierte horas hombre en la realización de esta tarea, el objeto de estudio del presente proyecto es poder optimizar dicho proceso.

1.2. FORMULACIÓN DEL PROBLEMA

De acuerdo a la problemática mencionada anteriormente se plantea lo siguiente: ¿Cuál es el modelo basado en Machine Learning que optimiza el pronóstico de ventas de la empresa Ricos Pan, año 2020 - 2021?

1.3. JUSTIFICACIÓN

Carranza & Sabría (2005) señalan que los pronósticos de ventas permiten medir la situación de la industria en el mercado y la participación de la empresa en ese mercado. Debido a que determina lo que realmente se puede vender en base a una realidad. Es así como el pronóstico de ventas es la proyección en el futuro de la demanda esperada dado un conjunto de restricciones ambientales. De este modo el sistema de pronósticos se configura como un sistema de aprendizaje. La medición de la performance de un pronóstico de ventas es porque a una mejor precisión esto impacta en diversos costos operativos; como: producción y logística; y en la satisfacción del cliente. Además, Cadena

Lozano (2016) menciona que Mentzer, Bienstock y Kahn, proponen una línea base para la gestión del pronóstico de ventas en las empresas, logrando identificar cuatro enfoques: el de independencia, enfoque concentrado, enfoque negociado y enfoque de consenso, además plantearon procedimientos que ayuden a mejorar continuamente los pronósticos de ventas e indicadores de calidad y exactitud, estos procedimientos consideran distintas etapas como:

a) Definición de las funciones de las áreas,
b) Definición del tipo de enfoque,
c) Determinación sobre cómo se desarrollan las tecnologías de información y comunicación para obtener los insumos y generar los pronósticos, y cómo estos pronósticos son articulados en las áreas funcionales,
d) Establecer una o varias medidas de desempeño o precisión del pronóstico.

También es importante señalar que la investigación de Mentzer & Moon (2005) se buscó identificar las buenas y malas prácticas de pronóstico en las empresas, concluyendo cuatro dimensiones: integración funcional, enfoque, sistemas y medidas de desempeño. Dentro de cada dimensión identificaron además cuatro etapas de comportamiento, cada etapa corresponde a una escala ordinal, es decir, a medida que aumenta el nivel de cada etapa se aumenta la eficiencia de los procesos. De esta manera podríamos identificar en qué dimensión se encuentra la empresa Ricos Pan con respecto a las buenas y malas prácticas de pronóstico, posicionándola en la dimensión de integración funcional y en la etapa 1, esto quiere decir, que la empresa, existe poca comunicación entre las áreas funcionales y cada área realiza su pronóstico de manera independiente, según sus necesidades y propósitos, no existe rendición de cuentas sobre la exactitud del pronóstico, esta situación podría explicar porque la empresa estaría realizando un incorrecto volumen de producción, lo cual se traduce en mermas o

desabastecimiento, los enfoques de dimensión y etapas descritos por Mentzer y Moon (2005) se dan bajo el concepto de gestión estrategia de pronóstico de acuerdo a Cadena Lozano (2016), dicho enfoque no se abordó en la presente investigación, el enfoque dado es el pronóstico de ventas en forma cuantitativa con los datos históricos que se dispone en la empresa.

Ahmed et al. (2010) indica que los modelos basados en Machine Learning en la última década están siendo serios contendientes a la estadística clásica en el área de pronóstico, por otro lado, Nojek et al. (2003) señala que "la elección e implementación de un método adecuado de pronósticos siempre ha sido un tema de gran importancia para las empresas. Se utilizan los pronósticos en el á rea de compras, marketing, ventas, etc. Un error significante en el pronóstico de ventas podría dejar a una empresa sin la materia prima o insumos necesarios para su producción, o podría generarle un inventario demasiado grande. En ambos casos, el pronóstico erróneo disminuye las utilidades de la empresa", de acuerdo a lo dicho anteriormente esto representa un problema para las empresas, más si están en el rubro de panadería como lo es Ricos Pan, ya que en este tipo de empresas los tiempos de vida del producto son cortos y la rotación es alta, es por ello que vemos una oportunidad de mejora en el proceso de pronóstico de ventas de la empresa.

1.4. OBJETIVOS

1.4.1. Objetivo general

Desarrollar un modelo basado en Machine Learning para optimizar el pronóstico de ventas de la empresa Ricos Pan, año 2020 - 2021.

1.4.2. Objetivos específicos

- Preprocesar los datos de venta que proporciona el Sistema de ventas de la empresa.
- Implementar los modelos basados en Machine Learning: redes neuronales convolucionales (CNN), redes neuronales recurrentes (LSTM y GRU) y máquina de soporte vectorial de regresión (SVR) para el pronóstico de ventas.
- Evaluar y determinar el modelo más óptimo basado en Machine Learning para el pronóstico de ventas.
- Comparar el pronóstico de ventas manual con el modelo basado en Machine Learning.

1.4.3. Limitaciones

La investigación se limita en analizar los datos presentados como muestra, ya que dichos datos fueron autorizados por la empresa Ricos Pan, por otro lado, el tiempo que demora en analizar cada producto es amplio e implica muchos recursos en hardware en tiempo de ejecución.

CAPITULO II

REVISIÓN DE LA LITERATURA

2.1. ANTECEDENTES DE LA INVESTIGACIÓN

Bontempi et al. (2013), su investigación titulada “Machine Learning Strategies for Time Series Forecasting” señala la creciente disponibilidad de datos históricos y la necesidad de realizar pronósticos precisos del comportamiento futuro, exige la definición de técnicas robustas y eficaces para poder inferir entre el pasado y futuro. El objetivo del estudio es presentar técnicas de Machine Learning centrándose en tres aspectos: formalización de problemas de pronóstico de un paso como tarea de aprendizaje supervisado, discusión de técnicas de aprendizaje local como una herramienta para tratar datos temporales y el papel de la estrategia de pronóstico de múltiples pasos. La investigación destaca el papel que cumple los pronósticos locales de aprendizaje frente a la no linealidad, la no estacionalidad y la acumulación de errores en la predicción de series de tiempo.

Ensafi et al. (2022), realizaron una investigación titulada “Time-series forecasting of seasonal items sales using machine learning – A comparative análisis”, donde investigaron un conjunto de datos públicos en donde se incluye las ventas de una tienda minorista de muebles, el objetivo de la investigación es pronosticar la venta de muebles a través de modelos basados en redes neuronales y clásicos, el estudio a su vez también realiza un proceso exploratorio y análisis de los datos presentados, al conjunto de datos aplican varios modelos de pronóstico, como SARIMA (método clásico), como también otros métodos de pronóstico avanzados basados en redes neuronales artificiales como Prophet, LSTM y CNN. Los resultados muestran la superioridad del método LSTM apilado sobre los otros métodos. Además, los resultados indican el buen rendimiento de

los modelos Prophet y CNN, los resultados fueron comparados utilizando métodos de medición como RMSE y MAPE.

Badal & Franzén (2019), en su investigación titulada “A Comparative Analysis of RNN and SVM”, hicieron un estudio con respecto al pronóstico del precio de la electricidad, los modelos fueron construidos y optimizados en un solo conjunto de datos históricos de un mercado de electricidad australiano donde los principales atributos de influencia son el precio, la demanda y el tiempo. El objetivo del estudio fue comparar dos métodos de aprendizaje automático una red neuronal recurrente RNN con LSTM, y SVM. El conjunto de datos entrenamiento y prueba se dividen en 80/20. Concluyen que las redes SVM superan a las redes RNN en cuanto a la precisión de la predicción, se observó diferencias y sensibilidad en los resultados debido a que se aplicaron una matriz de confusión.

Zhao & Wang (2017), realizaron una investigación titulada “Sales Forecast in E-commerce using Convolutional Neural Network”, señala que el pronóstico de ventas es una tarea esencial en el comercio electrónico y tiene un impacto crucial en la toma de decisiones comerciales informadas además señala que el pronóstico de ventas es un problema desafiante, ya que las ventas se ve afectadas por muchos factores, incluidas las actividades de promoción, los cambios de precios y las preferencias de los usuarios. El objetivo del estudio pretende dar un enfoque novedoso debido a que los nuevos métodos requieren de una ingeniería de características, tiempo y conocimientos de expertos, para superar esta limitación la investigación usa una red neuronal convolucional (CNN), probando en el conjunto de datos de www.cainiao.com, la red neuronal se alimenta con datos sin preprocesar ya que de acuerdo a este enfoque se puede extraer características de manera automática. Los resultados que obtuvieron validaron las expectativas de la dicha

investigación, probaron que dicha red puede identificar automáticamente características relevantes al conjunto de datos para el pronóstico de ventas.

Velásquez et al. (2010), en su investigación titulada "Predicción de series temporales usando máquinas de vectores de soporte", señala que las series de tiempo es un importante problema de investigación en ingeniería, economía, finanzas, economía y ciencias sociales, así mismo señala que las máquinas de vectores de soporte han sido utilizadas para predicción de series tiempo; como objetivo principal de esta investigación es proponer una técnica novedosa para estimar algunas constantes que son necesarias para el modelo SVM, y generalmente son establecidas de manera empírica, el conjunto de datos para medir el desempeño fueron conjunto de datos públicos, las series fueron: AIRLINE, POLLUTION, INTERNET, LYNX y PAPER. Concluyeron que a diferencia de otros investigadores los modelos basados en SVM solo fueron superiores solo en 3 de cada 5 casos, respecto a los modelos ARIMA y MLP.

Nojek et al. (2003), realizaron una investigación titulada "Pronóstico de ventas: comparación de predicción entre redes neuronales y método estadístico" cuyo objetivo consistió en estudiar la predicción de ventas mediante el uso de redes neuronales, comparar los resultados obtenidos con los pronósticos de un método estadístico clásico y establecer los entornos más adecuados para su uso. Los resultados de la investigación corroboraron que las redes neuronales realizaron un pronóstico más acertado con un 82.29% en comparación al método estadístico empleado y que este método requiere de un conocimiento en menor proporción comparado con los métodos estadísticos, sin embargo, es necesario que el usuario conozca la topología y el método de entrenamiento para cada caso en particular.

Augusto et al. (2014), en su investigación titulada “Comparación de predicción basada en redes neuronales contra métodos estadísticos en pronósticos de ventas” lleva a cabo una comparación y selección de un método para pronosticar las ventas de forma eficiente y así beneficiar a organizaciones que ofrecen sus productos al mercado, ello debido a que los pronósticos de ventas son datos de entrada a diferentes áreas de la empresa y la imprecisión de esta información podría desencadenar en gastos para la organización. En dicha investigación se trabajó con métodos y metodologías tales como Método de Hold Winters, la metodología Box Jenkins (ARIMA) y una Red Neuronal Artificial. Los resultados muestran que la red neuronal artificial obtuvo un mejor desempeño logrando el menor error cuadrático medio, de esta forma es posible establecer un panorama adecuado para el uso de la inteligencia artificial dentro de la industria. Parte de las conclusiones es que los métodos estadísticos para pronosticar parten de los datos históricos, en este caso en particular, el método más efectivo fue en la red neuronal artificial.

Osowski et al. (2004) en su investigación titulada “MLP and SVM Networks – a Comparative Study”, el objetivo del estudio es resumir y presentar la comparación de dos redes neuronales: multilayer perceptrón (MLP) and Support Vector Machine (SVM), los métodos mencionados se implementaron en Matlab, se analizó la complejidad y el tiempo de cálculo para la clasificación, predicción y regresión. Dando como resultado que en el modo de clasificación el invencible es SVM, mientras que en el modo de regresión la mejor habilidad de generalización posee MLP, la red MLP implementa una estrategia de aproximación global y emplea un pequeño número de neuronas ocultas a diferencia de SVM que se basa en una estrategia de aproximación local, en grandes cantidades de datos SVM, suele ser mucho más rápido.

2.2. MARCO TEÓRICO

2.2.1. Pronóstico de ventas

De acuerdo a Mentzer & Moon (2005) cualquier previsión de ventas debe considerarse como la mejor conjetura sobre la demanda de bienes y servicios de una empresa por parte de los clientes durante un horizonte concreto, dado un conjunto de supuestos por el entorno. Por otro lado, Anthony & Govindarajan (2011) indicaron que la forma de hacerlo depende si se utiliza un método cualitativo o un método cuantitativo, su objetivo es, con la mayor precisión posible tratar de predecir la cantidad de bienes se venderán y de este modo intentar disminuir los costes de inventario y transporte. Un pronóstico funciona como un sistema de gestión y tiene casi los mismos atributos que un presupuesto, aunque hay diferencias relevantes en ambos. Cuando se trata de la gestión de pronóstico de ventas es importante que se comprendan plenamente los fundamentos de la previsión de ventas se puede hacer una separación de varios aspectos y con ello obtener una imagen clara de lo que es la previsión de ventas, que indicadores afectan la previsión y lo más importante, que impactos tienen los errores en las previsiones (Aronsson & Jonsson, 2008).

2.2.2. Técnicas

Históricamente la demanda siguió patrones del tiempo y debido a ello, se han desarrollado enfoques estadísticos para identificar estos patrones que dirigen las previsiones de ventas (Mentzer & Moon, 2005). Los métodos utilizados pueden ser endógenos, que sólo utilizan las ventas históricas como entrada, o exógenos, que utilizan más variables que sólo los datos de las ventas históricas (Aronsson & Jonsson, 2008). De acuerdo a los autores citados las técnicas para pronóstico de ventas son:

- **Métodos de juicio**

  En contextos en donde los cambios grandes son normales los métodos de juicio con confiables ya que los datos históricos no son relevantes. Las desventajas de las personas a menudo son sesgadas y limitadas que vienen con soluciones internas de los gerentes, la precisión de este tipo de pronósticos es menor, que cuando se utiliza métodos estadísticos debido a sesgos y limitaciones.

- **Métodos de recuento**

  En este tipo de métodos el usuario pregunta a los clientes, competidores y otras personas sobre la percepción de un producto y al obtener datos, con suerte aumentaran la precisión del pronóstico de una empresa.

- **Series de temporales**

  Las series temporales se utilizan y desarrollan para identificar patrones de en datos históricos que se repiten en el tiempo.

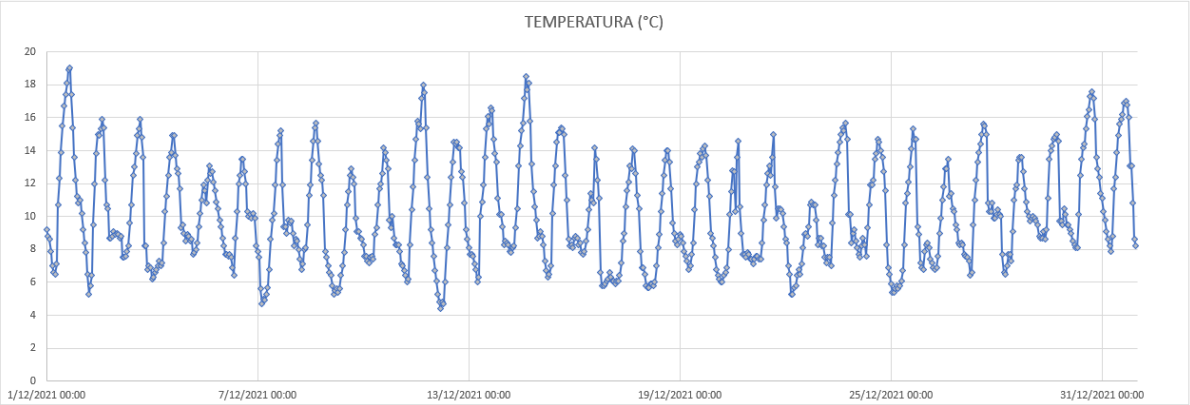
- **Métodos causales**

  Los métodos casuales esta diseñados para reconocer los patrones históricos que existen entre la demanda y las diferentes variables explicativas.

2.2.3. Series de tiempo

Las series de tiempo son una secuencia de mediciones históricas y de una variable observable en intervalos de tiempos iguales, las series de tiempo se estudian para diversos fines, entre ellos la previsión del futuro basado en el conocimiento del pasado (Bontempi et al., 2013). Los datos de la serie de tiempo son una valiosa fuente importante de información que se utilizan para la futura toma de decisiones, operaciones de planificación en diferentes industrias. En las últimas décadas el pronóstico basado en Machine Learning se convirtió en una herramienta muy popular en los sectores público y privado (Lazzeri, 2020).

Figura 1: Temperatura en la estación hidrométrica Puno.



Fuente: Adaptado de SENAMHI (2021).

En la Figura 1 se muestra la variación de temperatura de acuerdo a la estación hidrométrica ubicado en la Ciudad de Puno desde el 01 de diciembre del 2021 al 31 de diciembre del 2021, el intervalo de esta serie temporal es por hora y se muestra a través de marcas en la figura.

2.2.3.1. Descomposición de una serie de tiempo

Las series de tiempo se pueden descomponer en:

- **Componente Tendencia:** Una tendencia es un movimiento direccional consistente en una serie de tiempo, estas pueden ser deterministas o estocásticas, el primero se puede calcular, en caso de que la última sea aleatoria es poco probable que se pueda calcular (Tatsat et al., 2020). Considerando una línea de tendencia determinista en la ecuación 1.

$$m_t = a + \mu t \quad (1)$$

$$\nabla m_t = \mu \quad (2)$$

$$\nabla^2 m_t = 0 \quad (3)$$

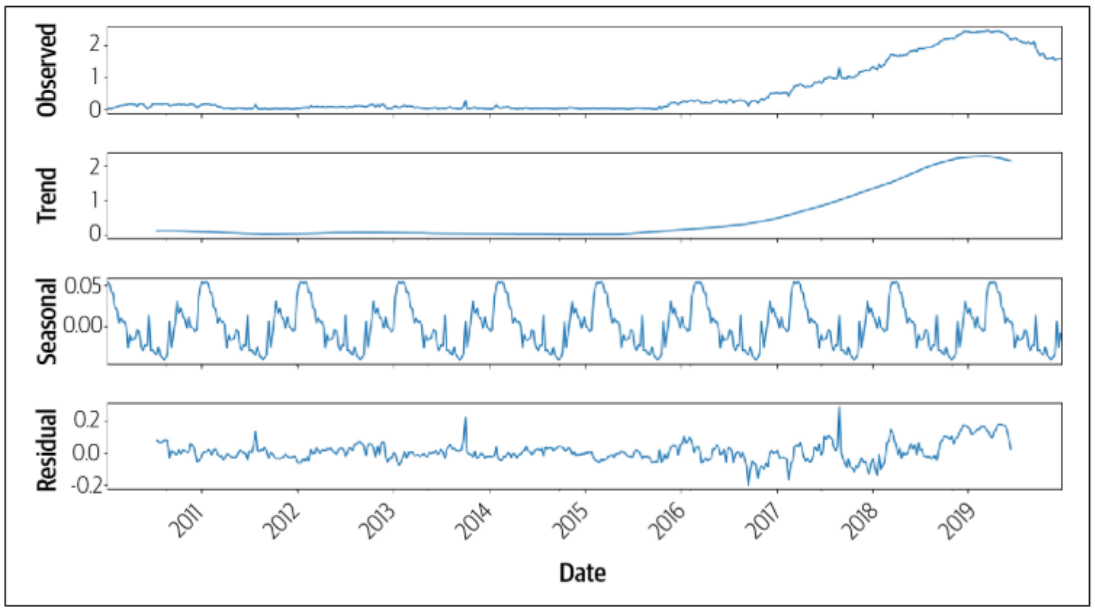
Donde m_t es la línea de tendencia, ∇m_t y $\nabla^2 m_t$ son las derivadas de la línea de tendencia y μ es la pendiente de la recta en las ecuaciones 2 y 3 respectivamente. Una tendencia móvil o estocástica no va satisfacer exactamente las ecuaciones antes mencionadas en cada periodo, en su lugar, se puede asumir que la tendencia tiene cierta perturbación en cada periodo, con una media cero y una pequeña varianza (Maravall, 1996).

- **Componente estacional:** Varias series de tiempo contienen variación estacional, esto particularmente se observa en series que representan venta o variaciones climáticas. Los componentes de la serie de tiempo se representar en la ecuación 4 (Tatsat et al., 2020).

$$y_t = S_t + T_t + R_t \tag{4}$$

Donde S_t es el componente estacional, T_t es el componente de tendencia, y R_t representa el residuo que no fue considerado en los dos componentes anteriores.

Figura 2: Serie de tiempo separada en sus componentes.



Fuente: Tatsat et al. (2020).

2.2.3.2. Clasificación de series de tiempo

De acuerdo a Palit, Ajoy K. & Popovic (2005), las series de tiempo se pueden clasificar dependiendo de las características de los datos:

- Estacionaria y no estacionaria
- Estacional y no estacional
- Linear y no linear
- Univariantes o multivariantes

2.2.3.2.1. Series de tiempo lineales

Las Series de tiempo lineales se generan a través de la observación de los procesos lineales, definidos en la ecuación 5.

$$y(t) = \sum_{j=-\infty}^{\infty} a_j x(t-j) \tag{5}$$

Donde $y(t)$ es el valor de la serie de tiempo en un tiempo determinado t, el coeficiente a_j está sujeta a la siguiente restricción:

$$\sum_{j=-\infty}^{\infty} |a_j| < \infty \tag{6}$$

2.2.3.2.2. Series de tiempo no lineales

Varias series de tiempo en ingeniería y macroeconomía requieren modelos no lineales. Algunas están representadas como series de tiempo bilineal, el modelo matemático está representado en la ecuación 7.

$$x_t = z_t + \sum_{i=1}^{p} a_i x_{t-i} + \sum_{j=1}^{q} b_j z_{t-j} + \sum_{i=1}^{r}\sum_{j=1}^{s} c_{ij} x_{t-i} z_{t-j} \tag{7}$$

Donde x_t, z_t es el valor de la serie de tiempo en un tiempo determinado t, $(a, b\ c)$ son constantes de la serie de tiempo bilineal.

2.2.3.2.3. Serie de tiempo univariante

Las series de tiempo univariante se caracterizan por obtener el muestreo de un solo patrón de observación, los valores como única variable física o de una única señal dependiente del tiempo en intervalos iguales.

2.2.3.2.4. Serie de tiempo multivariante

Las series temporales multivariante se generan por observación simultanea de dos o más procesos. Este tipo de series son muy comunes en la ingeniería, donde se muestran dos o más variables físicas por ejemplo temperatura, presión, flujo. Etc.

2.2.3.2.5. Serie de tiempo caóticas

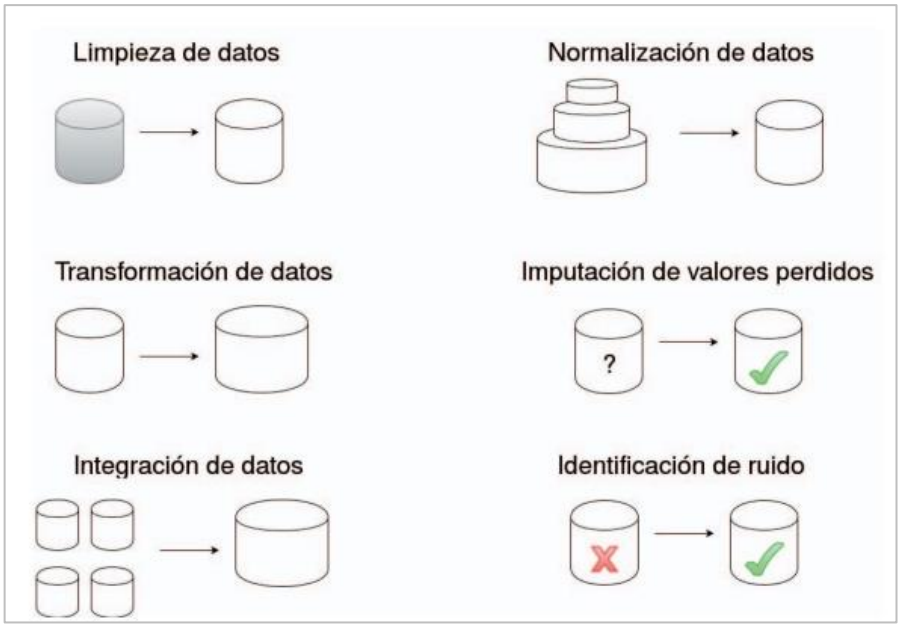
La característica principal de la serie de tiempo caótica, es que no tiene periodicidad definitiva, pueden tener valores que se pueden repetir al azar sin ninguna periodicidad definitiva.

2.2.4. Preprocesamiento y calidad de datos

La gran cantidad de datos que manejan las empresas, ha generado la necesidad de tener sistemas en los cuales confluya toda la información que es recopilada en fuentes de datos, como las bases de datos transaccionales; el preprocesamiento es una tarea necesaria para el análisis de datos, la justificación de este análisis de datos generalmente radica en que los datos vienen con ruido por diferentes razones (Hernández & Rodríguez, 2008). La eficacia de los algoritmos depende en gran medida de la calidad de datos, la cual puede ser garantizada por algoritmos de preprocesamiento, esta etapa es fundamental en el proceso de extracción de conocimiento, cuyo objetivo principal es obtener un conjunto

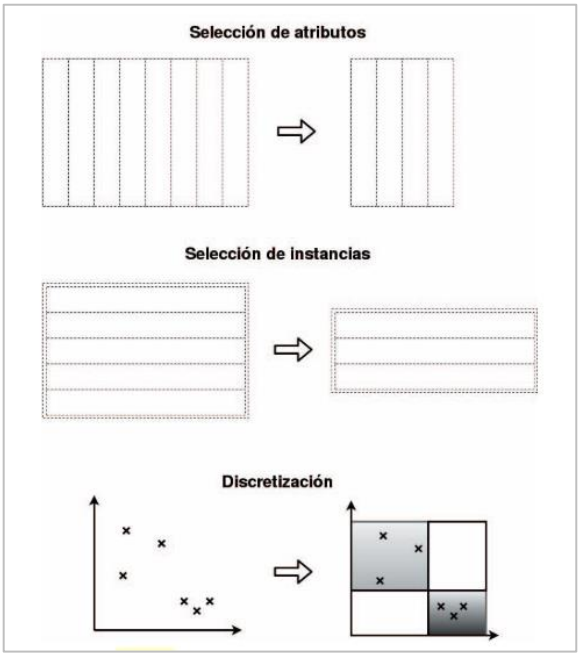
de datos con calidad para la fase de extracción de conocimiento. La preparación de datos como se muestra en la Figura 3 está dada por varias técnicas como limpieza de datos, normalización de datos, transformación de datos, imputación de valores perdidos, integración de datos, identificación de ruido, este tipo de técnicas son de uso obligatorio ya que sin ellas los algoritmos de extracción de conocimiento no podría, ejecutarse u ofrecerían datos erróneos, por otro lado de acuerdo a la Figura 4 tenemos las técnicas de reducción de datos que se orientan más a obtener una representación reducida de los datos originales manteniendo en gran medida la información obtenida en los datos originales, este tipo de técnicas solo se considerara cuando el tiempo de ejecución de los algoritmos son elevados (García et al., 2016).

Figura 3: Técnicas en preparación de datos.



Fuente: García et al. (2016).

Figura 4: Técnicas en reducción de datos.



Fuente: García et al. (2016).

2.2.4.1. Limpieza de datos

De acuerdo a KIM et al. (2002) la limpieza de datos se da por las siguientes razones: datos faltantes, datos no faltantes pero incorrectos, datos no faltantes y correctos. La tercera forma ocurre cuando se integran dos o más base de datos, las normas de representación no se usan de manera consistente. Algunos datos sucios se manifiestan de la combinación de formas mencionadas, por ejemplo, datos concatenados de manera errónea “Manuel Ponce” en lugar de “Ponce, Manuel”. En cuanto a la falta de datos debe ser completados, incluidos los nulos o la intervención de un experto en los datos que estamos analizando. Ye (2015) nos indica como debemos tratar con los datos faltantes además con los valores atípicos:

2.2.4.1.1. Tratar con los datos faltantes

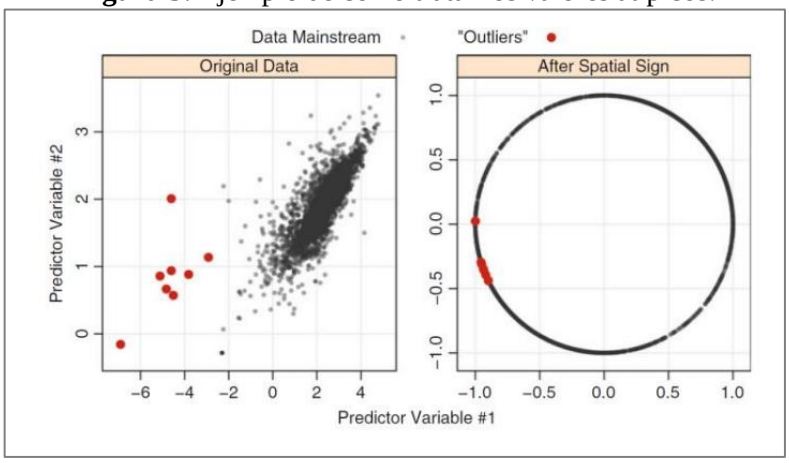
En general se pueden dividir en dos estrategias: el primero y el más simple es solo eliminar los datos faltantes directamente, si los datos se distribuyen de forma aleatoria, además esta acción va tener un impacto nulo en el análisis o algoritmo que se aplicara

después. La segunda estrategia es completar los datos faltantes de en función al resto de datos, se puede considerar el promedio para completar los valores que faltan o usar algoritmos bayesianos o arboles de decisiones para predecir los datos faltantes (Ye, 2015).

2.2.4.1.2. Tratar los valores atípicos

Un valor atípico se define como un punto de observación que dista de los datos generales. La presencia de estos valores atípicos puede romper la capacidad de análisis de los modelos, por ejemplo, existe un fuerte impacto al escalar los datos o normalizar estos. Identificar valores atípicos es una tarea difícil, una de las primeras formas es la visualización de datos, al observar la figura podemos identificar valores sospechosos. La eliminación de los valores atípicos solo se puede tomar cuando existan razones válidas. En lugar de eliminar los valores atípicos, otra forma es transformar los datos para minimizar el efecto causado por los valores atípicos. La señal espacial propuesta por Serneels en 2006 transforma los datos en una nueva esfera como se muestra en la Figura 5 (Ye, 2015).

Figura 5: Ejemplo de cómo tratar los valores atípicos.



Fuente: Ye (2015).

2.2.4.2. Normalización de datos

Mucho de los datos recopilados puede no ser suficientemente útiles para los modelos de Machine Learning, por ello es común realizar una serie de pasos de

manipulación para transformar los datos originales en mejores datos para el modelo (García et al., 2015).

2.2.4.2.1. Normalización Min-Max

Esta normalización tiene como objetivo escalar todos los valores numéricos a un rango específico y se obtiene aplicando la ecuación 8.

$$v' = new - min_A + R\left(\frac{v - min_A}{max_A - min_A}\right) \tag{8}$$

Donde v' es el nuevo valor normalizado, v es el valor a transformar, el rango es especificado por R y $[new - min_A, new - max_A]$, max_A min_A son los valores máximo y mínimo respectivamente.

2.2.4.2.2. Normalización Z-score

En algunos casos, la normalización min max no se puede aplicar, cuando no se conocen valores mínimos o máximos, o talvez se tiene los valores mínimos y máximos, pero son inviables debido a la presencia de valores atípicos que pueden sesgar la normalización. La normalización Z-Score está dada por:

$$v' = \frac{v - \bar{A}}{\sigma_A} \tag{9}$$

Donde v' es el nuevo valor normalizado, v es el valor sin normalizar, $\bar{A}$ es la media de los valores, σ_A es la desviación estándar.

2.2.4.3. Transformación de datos

Ye (2015) indica que el procesamiento de datos implica la transformación de los datos en una forma adecuada para el análisis, las técnicas más básicas y directas son: el centrado y la escala de datos. Muchas técnicas de minería de datos como la distancia euclidiana, requieren que los datos se centren y se escalen antes de ingresar al modelo,

este tipo de transformaciones ayudan a mejorar la interpretabilidad de las estimaciones de parámetros cuando hay interacción con el modelo. García et al. (2015) considera que la transformación de datos generalmente combina atributos usando diferentes fórmulas matemáticas:

2.2.4.3.1. Transformación lineal

En el área de descubrimientos científicos y control de máquinas, las normalizaciones no pueden ser suficientes para adaptar datos para mejorar el modelo generado. En estos casos se puede agregar la información contenida en varios atributos y puede ser beneficiosa, se basan en transformaciones algebraicas simples, como sumas, promedios, rotaciones, traslaciones, etc. Tenemos $A = A1, A2, \dots, An$ y un conjunto de atributos, $B = B1, B2, \dots, Bm$, de A, si la siguiente expresión es aplicada (García et al., 2015):

$$Z = r_1 B_1 + r_2 B_2 + \cdots + r_m B_m \tag{10}$$

Un atributo derivado es construido tomar una combinación lineal de atributos en B, un caso especial es cuando $r1 = r2 = \cdots = r * m = 1/m$, esta situación promedia los atributos considerados en B.

$$Z = (B_1 + B_2 + \cdots + B_m)/m \tag{11}$$

2.2.4.3.2. Transformación cuadrática

Las transformaciones cuadráticas se crea un nuevo atributo de la siguiente manera:

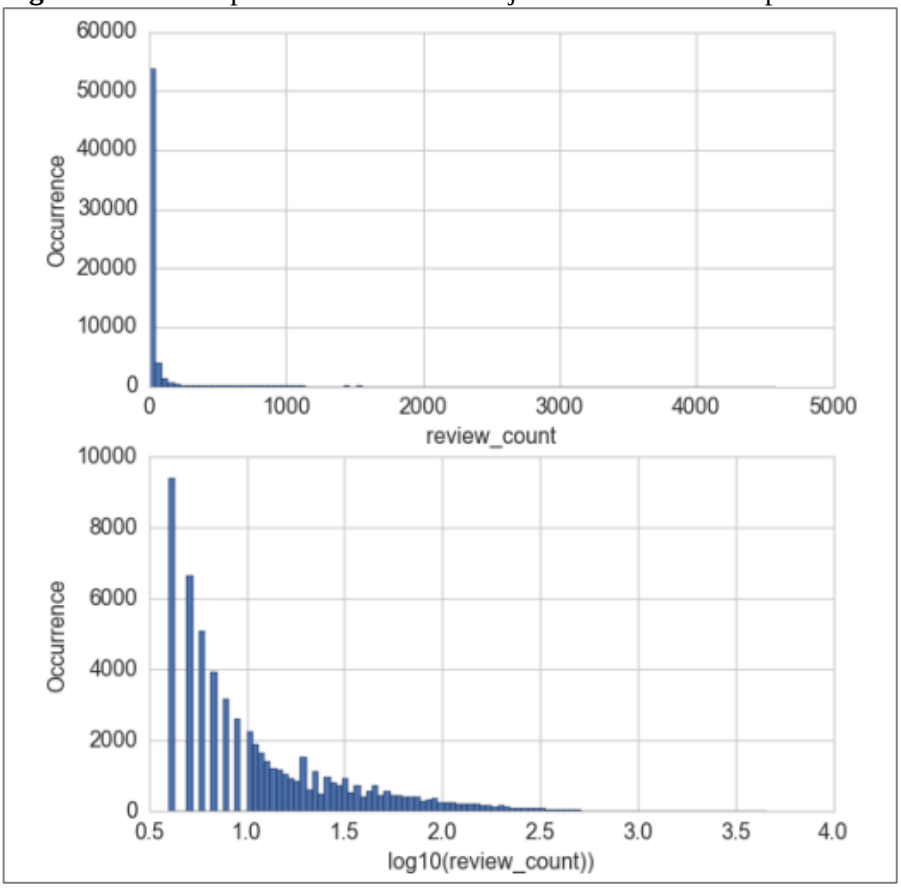
$$Z = r_{1,1} B_1^2 + r_{1,1} B_1^2 + \cdots + r_{m-1,m} B_{m-1} B_m + r_{m,m} B_m^2 \tag{12}$$

Donde Z es el valor transformado, B_n son atributos, $r_{i,j}$ es un numero real. Este tipo de transformaciones se estudiaron a fondo y pueden ayudar a transformar datos para que sean separables (García et al., 2015).

2.2.4.3.3. Transformación logarítmica

Según Zheng & Casari (2018) la función logaritmo es la inversa de la funcione exponencial se define de la siguiente manera $log_a(a^x) = x$, donde a es una constante positiva y x puede ser cualquier número. La función $log_{10}(x)$ mapea el rango de [1,10] a [0,1], [10,100] a [1,2], en otras palabras, comprime el rango de números elevados y expande el rango de números pequeños. En la Figura 6 compara los histogramas del conjunto de datos del número reseñas dejadas por el usuario a negocios (del conjunto de datos Yelp Bussines) antes y después de la transformación.

Figura 6: Reseñas por los usuarios del conjunto de los datos Yelp Bussines.



Fuente: Zheng & Casari (2018).

2.2.4.3.4. Diferencia

Las transformaciones como los logaritmos pueden ayudar a estabilizar la varianza de una serie temporal. La diferencia puede ayudar a estabilizar la media de una serie temporal eliminando los cambios en el nivel de una serie temporal y, por lo tanto, eliminando (o reduciendo) la tendencia y la estacionalidad (García, 2022). De acuerdo a IBM (2022) existen dos niveles de diferenciación.

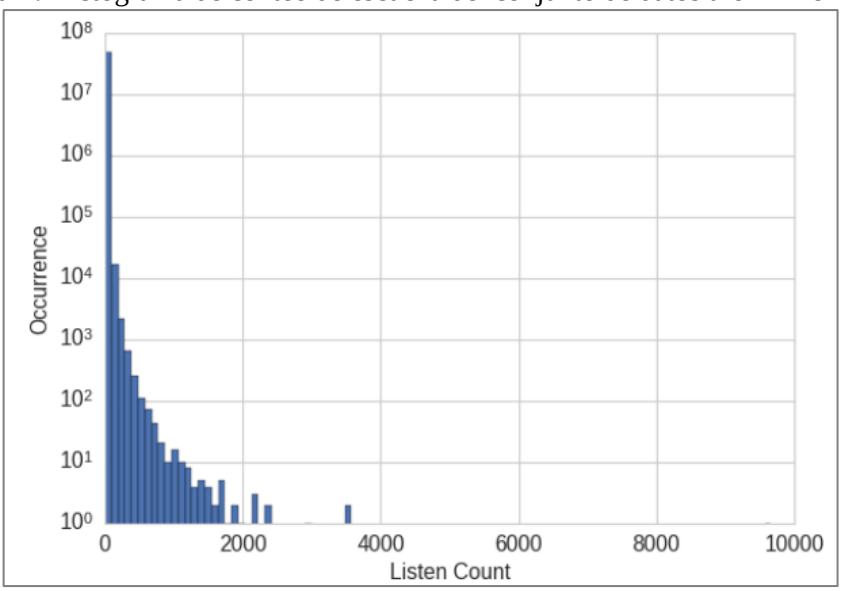
- **Diferenciación simple.** Se calculan las diferencias existentes entre cada valor y el anterior de la serie, a excepción del valor más antiguo de la serie. Por tanto, la serie diferenciada tendrá un valor menos que la serie original.
- **Diferenciación estacional.** Es idéntica a la diferenciación simple, excepto en que se calculan las diferencias existentes entre cada valor y el valor estacional.

2.2.4.4. Binarización de datos

De acuerdo a Zheng & Casari (2018), en la era del Bigdata, la cantidad de datos se puede acumular rápidamente sin límite. Cuando los datos se pueden producir en grandes volúmenes y velocidades, es muy probable que contengan algunos valores extremos, una buena idea es verificar la escala y determinar si mantener los datos crudos o convertirlos en valores binarios para indicar la presencia o no, o convertirlos en otra granularidad. Para ilustrar estas ideas veamos algunos ejemplos: Supongamos que nuestra tarea es construir un sistema de recomendación de música para usuarios. Un componente de la recomendación podría predecir cuanto le gusta una canción particular a un usuario. Dado que los datos contienen los recuentos reales de escucha de las canciones, ¿Debería ser el objetivo de la predicción? Esto sería correcto pensando que si un gran umero de escucha significa que realmente le gusta la canción y un recuento bajo significa que no están interesados en ello. Sin embargo, los datos muestran que el 99% de las escuchas son 24 o menos, también hay algunos conteos escuchados en miles como máximo 9 667.

Como se muestra la Figura 7, los picos del histograma están cerca de cero, pero más de 10 000 tienen mayores recuentos, estos valores son anómalamente grandes, si tuviéramos que intentar predecir aspectos reales de escucha, se deberían retirar estos valores grandes.

Figura 7: Histograma de conteo de escucha del conjunto de datos the Million Song.



Fuente: Zheng & Casari (2018).

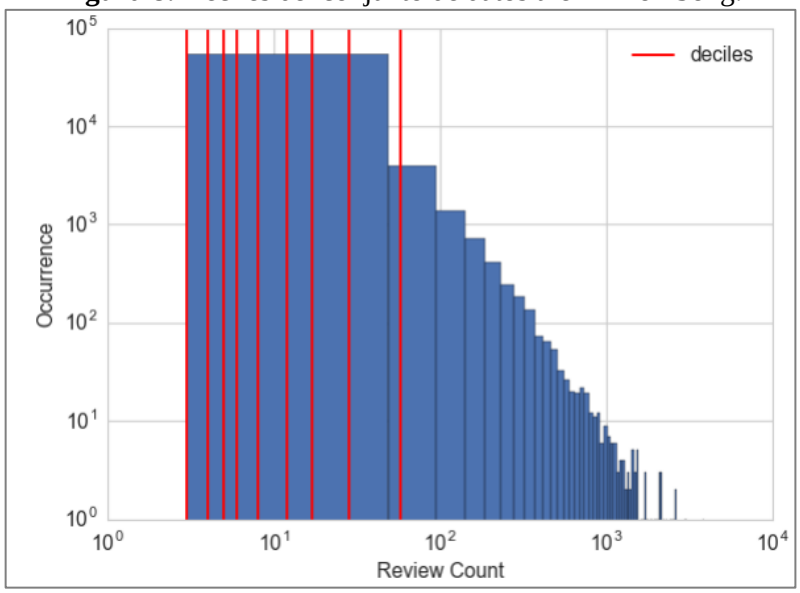
En el conjunto de datos Million Song, el recuento de escuchas, no es una medida robusta. Los usuarios tienen diferentes hábitos de escucha, algunas personas ponen sus canciones favoritas en un bucle infinito, mientras que otros suelen ponerlo solo en ocasiones especiales. No podemos decir que alguien que escucha una canción 20 veces, debe gustarle el doble de otra persona que lo escucha 10 veces. Una representación más robusta del usuario es binarizar el recuento y el clip, todos cuenta más de 1 a 1 como se muestra en el siguiente código, en otras palabras, si el usuario escucho una canción al menos 1 vez lo consideramos como gusto por parte del usuario (Zheng & Casari, 2018).

```
>>> import pandas as pd
>>> listen_count = pd.read_csv('millionsong/train_triplets.txt.zip',
...                            header=None, delimiter='\t')
# The table contains user-song-count triplets. Only nonzero counts are
# included. Hence, to binarize the count, we just need to set the entire
# count column to 1.
>>> listen_count[2] = 1
```

2.2.4.5. Cuantificación datos

Los cuantiles son valores que dividen los datos en porciones iguales. Por ejemplo, la mediana divide a los datos en mitades, la mitad de los puntos son más pequeños y la otra mitad más grandes que la mediana. Los cuartiles dividen a los datos en cuartos, deciles en décimas, etc. En el siguiente ejemplo se demuestra cómo se calcular los deciles para la cantidad de reseñas dejadas por los usuarios en negocios (conjunto de datos Yelp Bussines Review), y la Figura 8 muestra como son los deciles en el histograma, esto muestra un panorama más limpio acerca del sesgo hacia los conteos de las reseñas más pequeños (Zheng & Casari, 2018).

Figura 8: Deciles del conjunto de datos the Million Song.



Fuente: Zheng & Casari (2018).

2.2.5. Pronóstico de series de tiempo

2.2.5.1. Métodos clásicos

2.2.5.1.1. Suavizado exponencial

El suavizado exponencial es particularmente para pronósticos de corto plazo, emplea factores de ponderación para valores pasados, los factores decaen

exponencialmente con la distancia de los valores pasados con los valores actuales. Esto permite una formulación compacta de algoritmo de pronóstico en el que solo se requieren algunos datos recientes y se necesita menos cálculos, esto es altamente relevante para las aplicaciones de automatización industrial, donde se utilizan controladores programables y procesadores de señales (Palit, Ajoy K., Popovic, 2005). En el método de suavizado exponencial se utiliza el concepto de promedio móvil dada por la siguiente relación:

$$x_e(t) = ax(t) + (1 - a)x_e(t - 1) \tag{13}$$

Con el pronostico

$$\Phi(t + k) = x_e(t + k) \tag{14}$$

Donde $x_e(t)$ es el valor suavizado exponencialmente, $x(t)$ es el valor observado en el mismo punto del tiempo, a es el valor de la constante de suavización, $x_e(t - 1)$ es el anterior valor suavizado exponencialmente, y $\Phi(t + k)$ es el valor pronosticado de acuerdo al tiempo $t + k$.

De acuerdo a Carbonero Pinto Carbonero Pinto (2021, p. 6) este método se utiliza para “suavizar dicha serie temporal con el objetivo de reducir las fluctuaciones y poder ver una tendencia que a simple vista no es clara”.

2.2.5.1.2. ARIMA

El método Arima puede ser adecuada para la mayoría de series de tiempo regulares que exhiban un comportamiento sistemático sin perturbaciones aleatorias que sean pequeñas en comparación con sus componentes sistemáticos, no son apropiados para modelar las series de tiempo que tienen un comportamiento cíclico irregular, por ejemplo: los huracanes, inundaciones, terremotos, etc (Chatfield, 2005).

De acuerdo a Tatsat et al. (2020) la mayoría de moldeos de serie de tiempo tiene como objetivo incorporar las tendencias, la estacionalidad, y el resto, al mismo tiempo aborda la autocorrelación, una de los modelos más utilizados es el modelo $ARIMA$.

- $AR_{(p)}$: representa la auto regresión es decir la regresión de la serie de tiempo en sí misma, los valores actuales dependen de los valores anteriores con algún retraso, el retraso máximo se conoce como p
- $I(d)$: representa el orden de integración.
- $MA_{(q)}$: representa el promedio móvil.

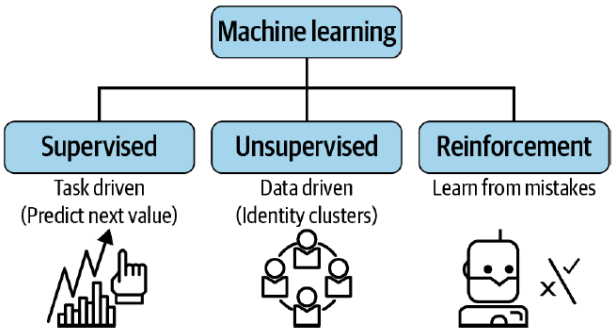
2.2.6. Métodos de pronóstico Machine Learning

2.2.6.1. Introducción

El sueño que las maquinas podría aprender es tan antiguo como las computadoras, tal vez aún más antiguas, durante mucho tiempo permanecieron como un sueño. El perceptrón de Rosenblatt desencadeno una ola de actividad, pero al poco tiempo solo fue de corta duración, sin compañías de software, ninguna investigación importante de seguimiento, el aprendizaje de las maquinas fue condenado a vivir a la sombra de las disciplinas más exitosas (Kubat, 2017). El Machine Learning es un tipo de inteligencia artificial, que provee a los sistemas la habilidad de aprender de la experiencia, sin la necesidad de programarla exactamente para ese propósito. Cuando se trabaja con Machine Learning se tiene que analizar grandes cantidades de datos, con el objetivo de entrenar modelos (Sanders, 2019). El aprendizaje de maquina (Machine learning) se define como un proceso automatizado que extrae patrones de datos para construir modelos utilizados en aplicaciones de datos predictivos (Kelleher et al., 2015) en el presente trabajo se usara el aprendizaje supervisado para poder construir modelos y luego hacer pronósticos en base a estos. Los algoritmos de Machine learning se pueden

clasificar en aprendizaje supervisado, no supervisado, reforzado como se muestra en la Figura 9.

Figura 9: Tipos de aprendizaje.



Fuente: Tatsat et al. (2020)

- **Aprendizaje supervisado:** Es un tipo de aprendizaje supervisado es un tipo de sistema de aprendizaje donde se proveen tanto las entradas como las salidas deseadas, los datos se identifican a priori para ser proporcionados al algoritmo de aprendizaje para el manejo futuro de los datos. El objetivo del aprendizaje supervisado es estudiar muchos datos anteriores para poder hacer predicciones sobre datos futuros, se llama aprendizaje supervisado porque los científicos de datos supervisan el algoritmo de aprendizaje. El proceso se basa en los siguientes pasos (Bontempi et al., 2013):
  1. Alimentar los datos al algoritmo, incluyendo características adicionales al modelo.
  2. Usar estos datos para entrenar al modelo
  3. Se prueba e implementa el modelo
  4. Usar el modelo desplegado, alimentando de datos nuevos y recibiendo las predicciones retornadas por el modelo.
- **Aprendizaje no supervisado:** Sería un gran error que el Machine Learning requiera siempre ejemplos de salida, los datos de entrada no siempre tienen un dato de salida o la salida no se conoce, la información útil puede ser tomada incluso si el resultado

no es conocido. Esto a veces se llama aprendizaje no supervisado, en contraste del aprendizaje supervisado, el aprendizaje supervisado se enfoca en descubrir propiedades útiles de los datos disponibles (Kubat, 2017).

- **Aprendizaje reforzado:** Este tipo de aprendizaje se basa en experiencias, recompensas y castigos, es el concepto central detrás del aprendizaje de refuerzo, se trata de tomar acciones adecuadas para maximizar las recompensas en determinada situación (Tatsat et al., 2020).

2.2.6.2. Como diseñar un pronóstico

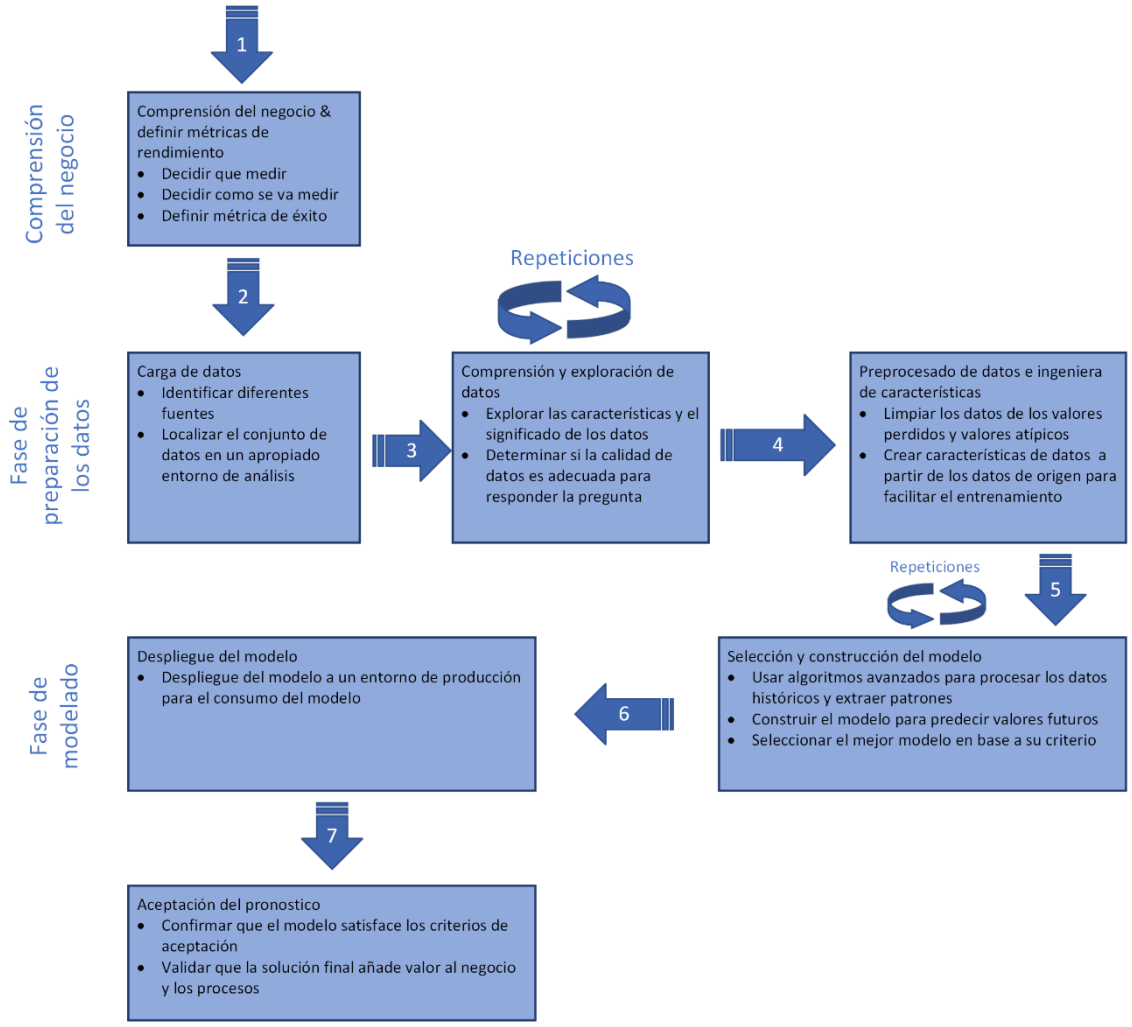
Lazzeri (2020) presenta algunas técnicas para la construcción de soluciones para un pronóstico de series de tiempo siguiendo la secuencia mostrada en la Figura 10 usando algoritmos de Machine Learning.

- **Entender y definir las métricas de rendimiento:** en este paso se refiere más a un aspecto empresarial en donde debemos comprender y considerar las ventajas y desventajas de implementar un pronóstico de tiempo. Antes de realizar una inversión, se debe garantizar que el análisis predictivo y el Machine Learning sean efectivos y aplicables.
- **Obtener datos:** es el proceso de recopilación e importación de datos para poder ser limpiados y analizados o guardados en una base de datos.
- **Exploración y análisis de datos:** Una vez que los datos sin procesar fueron guardados en forma seguro, están listos para ser explorados y transformados en una forma que se puedan limpiar y aplicar la ingeniería de características.
- **Pre procesado de datos e ingeniería de características:** en este paso es donde se limpian los datos de algunos valores atípicos, completan algunos datos faltantes y se

crean características adicionales con datos en bruto para alimentar los modelos de Machine Learning.

- **Construcción y selección del modelo:** La fase del modelado es donde los datos se convierten en un modelo, el núcleo de este proceso hay algoritmos avanzados que escanean los datos históricos (datos de entrenamiento), extrae patrones y construyen modelos. Luego este modelo puede usarse para predecir nuevos datos que no se utilizaron en la fase del entrenamiento.
- **Implementación del modelo:** En este paso consiste en la integración del modelo de Machine Learning, a un entorno de producción para comenzar a utilizarlo, es una de las últimas etapas del aprendizaje máquina.
- **Aceptación de la solución de pronóstico:** Es el último paso en la implementación, los científicos de datos deben confirmar y validar los resultados del modelo, y también se debe validar el modelo en el entorno de producción y satisfacer los criterios de rendimiento.

Figura 10: Pasos para diseñar un pronóstico de series de tiempo.



Fuente: Adaptado de Lazzeri (2020).

2.2.6.3. Máquinas de soporte vectorial (SVM)

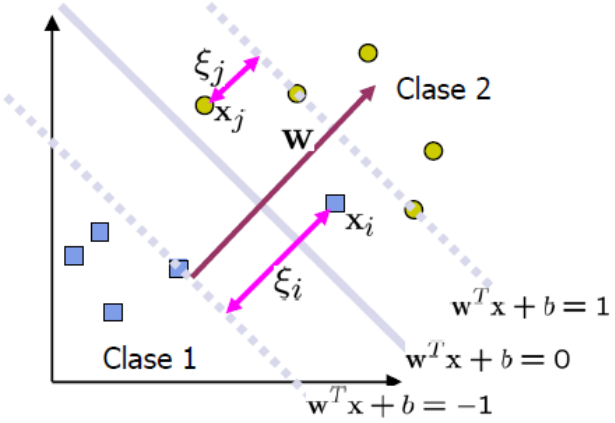
Este método forma parte de aprendizaje supervisado, “las máquinas de vectores de soporte es un tipo de red neuronal que fue originalmente diseñada para la solución de problemas no lineales de clasificación, pero recientemente se ha aplicado a problemas de regresión y predicción de series temporales.” (Velásquez et al., 2010, p. 2), por otro lado Parrella (2007) indica que las máquinas de vectores de soporte son una rama del aprendizaje estadístico parte de una serie de ejemplos para crear “un tomador de decisiones” que intenta predecir nuevos valores y puede ser subdivido en dos partes:

- **Aprendizaje:** Consiste en el entrenamiento del SVM con ejemplos que tiene a su disposición
- **Predicción:** Donde nuevos datos son insertados, en partes donde no se conocía los datos.

2.2.6.3.1. Modo de clasificación

De acuerdo a Betancourt (2005) considera dos casos, caso linealmente separable y no linealmente separable, se considera que el caso lineal solo es una consecuencia del caso no lineal. En la mayoría de casos la solución es mapear el espacio de entrada en un espacio de características de una dimensión mayor y buscar el hiperplano optimo allí en la Figura 11 se muestra una representación gráfica.

Figura 11: Maquinas de soporte vectorial, caso no linealmente separable.



Fuente: Betancourt (2005).

En general para tratar con datos no linealmente separables el análisis previo puede ser generalizado introduciendo algunas variables no negativas de modo que la ecuación que define el hiperplano quedaría definida de la siguiente manera:

$$\min\left\{\frac{1}{2}w\cdot w + C\sum_{i=1}^{l}\xi_i\right\} \tag{15}$$

$$s.a. \quad y_i(w\cdot z_i + b) \geq 1-\xi_i\,, i=1,\ldots,l$$

$$\xi_i \geq 0,$$

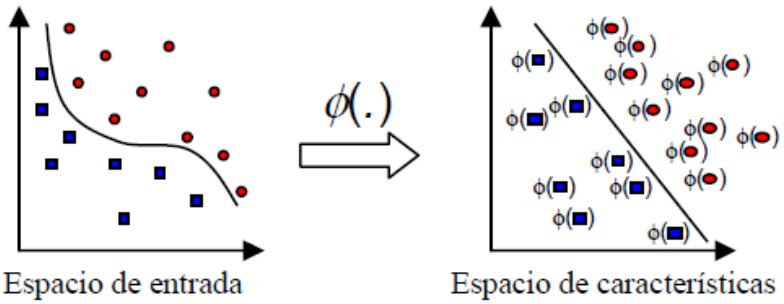
Donde C es una constante y puede ser definido como un parámetro de regularización, w pertenece a los reales, ξ_i puede ser tomado como algún tipo de medida de error en la clasificación, y_i pertenece a los puntos etiquetados (x_i, y_i) de la clasificación.

De otro lado Parrella (2007) indica una notación equivalente para la solución de SVM.

$$f(x) = sign(\sum_{i=1}^{l} y_i \alpha_i (x \cdot x_i) + b) \tag{16}$$

También indica que sustituimos el termino $(x \cdot x_i)$ con otra función llamada kernel esto permite que el crecimiento la solución de SVM, que comienza a crear otro espacio y ese "otro espacio" se llama espacio de características. Esto nos permite cambiar la información de un espacio lineal a otro. $sign$ se refiere a la función Signo, como se muestra en la Figura 12:

Figura 12: Idea de uso de kernel para la transformación del espacio de los datos.



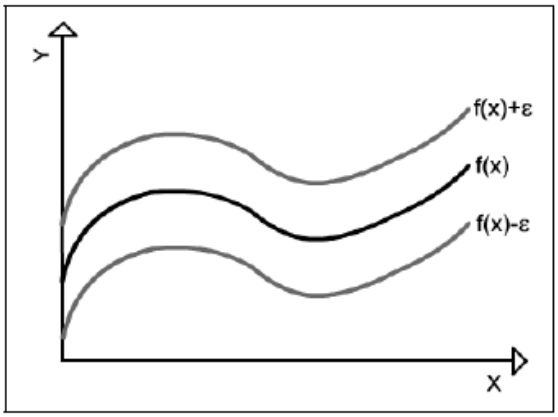
Fuente: Betancourt (2005).

2.2.6.3.2. Modo de regresión

Para Parrella (2007) el modo de regresión SVR usa los mismos principios que el modo de clasificación, con algunas diferencias menores. En el caso de regresión, existe un margen de tolerancia definido en el problema del modo de clasificación. Se podría

definir una línea que se convierte en un "tubo" tolerante a los errores de acuerdo a la Figura 13.

Figura 13: Tubo SVR.



Fuente: Parrella (2007).

Regresando al enfoque matemático, el problema de optimización cuadrática se convierte en:

$$min\frac{1}{2}\boldsymbol{w}^T\cdot\boldsymbol{w} \qquad (17)$$
$$s.t.\begin{cases} y_i-(\boldsymbol{w}^T\cdot\phi(x)+b)\leq\in \\ (\boldsymbol{w}^T\cdot\phi(x)+b)-y_i\leq\in \end{cases}$$

Donde $\phi(x)$ es la función kernel mencionada en el numeral anterior, **w** es el margen de la dupla (x_i, y_i) de los datos de entrenamiento. Adicionalmente se puede agregar un limite para establecer la tolerancia en el número de errores:

$$\min\frac{1}{2}\boldsymbol{w}^T\cdot\boldsymbol{w}+C\sum_{i=1}^{l}(\xi_i+\xi_i^*) \qquad (18)$$
$$s.t.\begin{cases} y_i-(\boldsymbol{w}^T\cdot\phi(x)+b)\leq\in+\xi_i \\ (\boldsymbol{w}^T\cdot\phi(x)+b)-y_i\leq\in+\xi_i \\ \xi_i,\xi_i^*\geq 0, i=1\ldots l \end{cases}$$

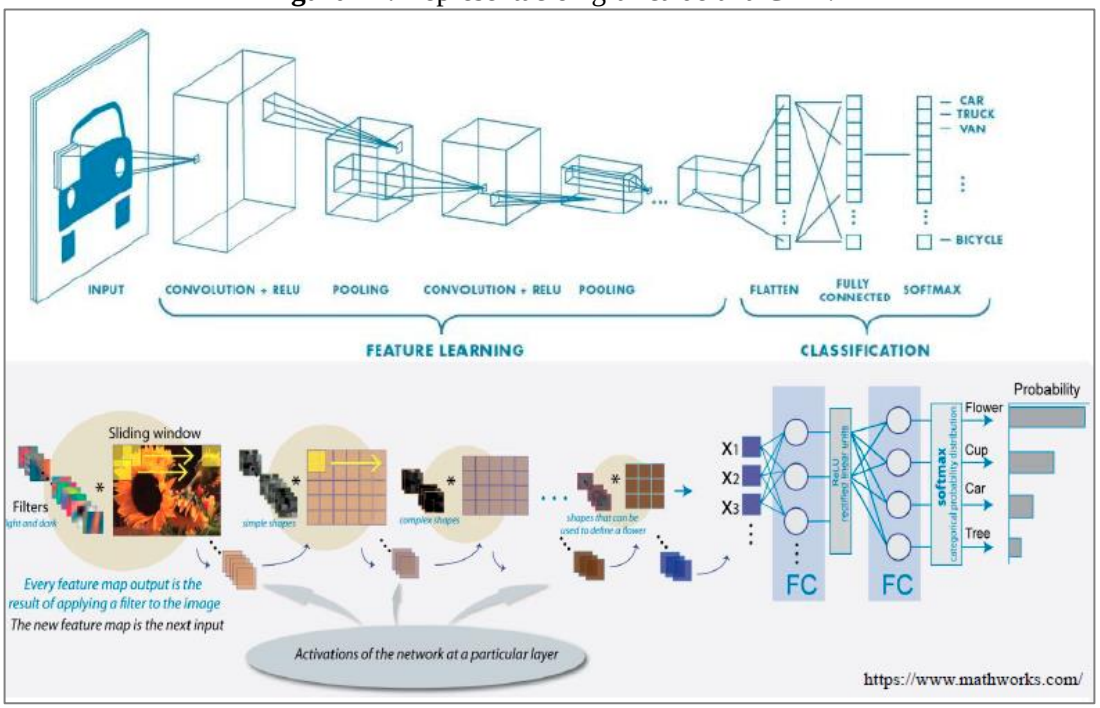
Donde $\boldsymbol{w}^T$ se refiere a la función transpuesta, $\in$ y ξ_i son parámetros que definen la máxima tolerancia. El principio es similar para SVM, una vez entrenado, el SVR puede generar predicciones de acuerdo a la siguiente formula:

$$f(x) = \sum_{i=1}^{l} \theta_i \phi(x, x_i) + b \tag{19}$$

2.2.6.4. Redes neuronales convolucionales (CNN)

Una red neuronal convolucional ha tenido resultados sobresalientes en la anterior década en una variedad de campos relacionados, desde el procesamiento de patrones, procesamiento de imágenes hasta el reconocimiento de voz, uno de los aspectos más importantes es la reducción de parámetros utilizados en las redes neuronales artificiales clásicas. Un ejemplo claro es que cuando solicitamos una detección de rostro, no se necesita saber la posición del rostro solo nos preocupamos de la detección independientemente de su posición en las imágenes dadas (Albawi et al., 2018).

Figura 14: Representación gráfica de una CNN.

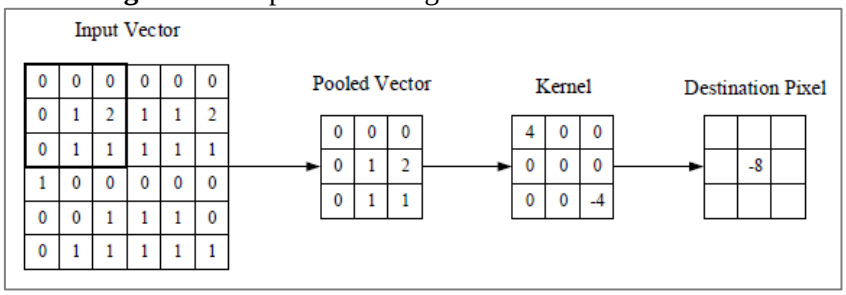


Fuente: Kim (2017).

- **Convolución:** Asumamos que cada neurona recibe un pixel de la imagen de entrada de 32x32x3 (generalmente se divide en 3 por los canales RGB) para intentar conectar a una capa oculta, necesitamos otros 32x32x3 de neuronas, en total necesitaríamos

más de 6000 neuronas para ajustar realizar la conexión. Claramente este no es un método eficiente, para eso tenemos que usar otro enfoque, podemos conectar la siguiente capa oculta con solo 5x5x3 neuronas, el número de conexiones se reducen drásticamente, por otro lado, la fijación de pesos para las conexiones es similar a deslizar una venta de 5x5x3 en las neuronas de entrada y mapeando la salida generada en el lugar correspondiente, una representación gráfica podemos observar en la Figura 15. Esto nos brinda la oportunidad de reconocer características independientes de sus pociones en la imagen. Y esta es la razón por la que se llaman convoluciones (Albawi et al., 2018).

Figura 15: Representación gráfica de una convolución.



Fuente: O'Shea & Nash (2015).

- **CNN y las series de tiempo:** Desde que AlexNet ganara la competición ImageNet en 2012, las CNN han visto muchas aplicaciones en diferentes ámbitos, motivados por este éxito los investigadores han comenzado a adoptarlos para el análisis de tiempo. La convolución se puede ver como aplicando y deslizando un filtro (kernel) sobre la serie de tiempo, a diferencias de las imágenes el filtro solo tiene una dimensión, este tambien puede verse como una transformación genérica no lineal de una serie de tiempo. Una forma general de aplicación de la convolución para un tiempo determinado t, se da en la siguiente relación (Ismail Fawaz et al., 2019):

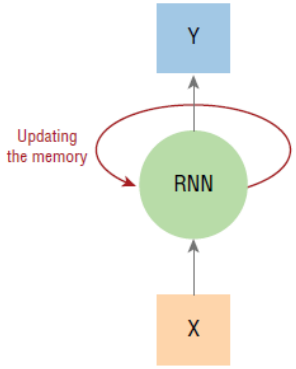
$$C_t = f\left(w * X_{t-\frac{l}{2}:t+\frac{l}{2}} + b\right) \mid \forall t \epsilon [1, T] \tag{20}$$

Donde C denota el resultado de la convolución (produto punto *) aplicado a la serie de tiempo invariada X de longitud T con un filtro w, b como parámetro bias, y la función f como una ReLU (Rectified Linear Unit)

2.2.6.5. Redes neuronales recurrentes (RNN)

Las redes neuronales recurrentes se crearon el 1980, pero en la actualidad ha estado ganando popularidad debido al aumento de la capacidad computacional, además en los últimos años, la unidad de procesamiento gráfica (GPU) crece más rápido que la CPU (Luo et al., 2005) , este tipo de redes son especialmente útiles con datos secuenciales porque cada neurona puede usar una memoria interna para mantener información sobre su entrada anterior. Este tipo de red neuronal tiene un problema fundamental de no poder capturar dependencias a largo plazo en una secuencia, esta fue una de las principales razones por la cual las RNN desaparecieron en la práctica (Lazzeri, 2020). Existe un tipo especial de RNN, las redes LSTM este tipo de redes son diseñados para trabajar en dependencias a largo plazo en una secuencia de datos, además existe una unidad recurrente cerrada GRU que también estan diseñados para dependencias a largo plazo y funcionan bien con datos secuenciales (Shewalkar et al., 2019).

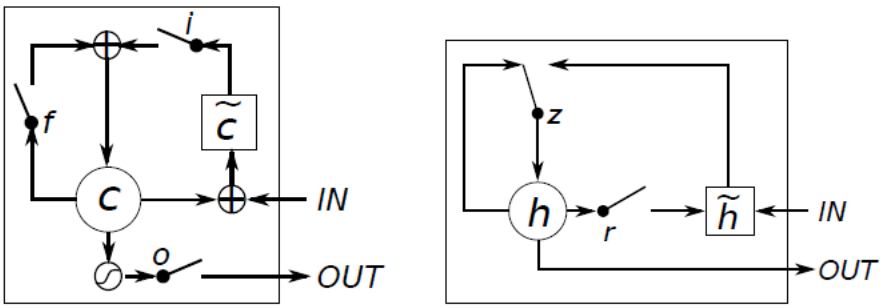
Figura 16: Representación de una Red Neuronal Recurrente (RNN).



Fuente: Lazzeri (2020).

Este tipo de redes muestran éxito en varias aplicaciones que involucran datos secuenciales o temporales, por ejemplo, se han aplicado en reconocimiento de voz, procesamiento de lenguaje natural, traducción automática. Las variaciones LSTM y GRU, fueron recientemente introducidas has demostrado que funcionan con éxito en secuencias largas. Este éxito se debe principalmente a las señales de la red de compuertas que controlan como se maneja la presente entrada y la memoria previa, para actualizar la activación actual y producir el estado actual. Estas puertas tienen su propio conjunto de pesos que se actualizan adaptativamente en la fase de entrenamiento, debido a esto se introduce un aumento en la parametrización de las redes de compuertas, que en consecuencia tienen un gasto computacional mayor con respecto a las RNN estándar; se observa que la variación LSTM usa 3 compuertas distintas, mientras que GRU se reduce a 2 compuertas como se muestra en la Figura 17.

Figura 17: Representación de las variaciones de una red neuronal recurrente LSTM y GRU.



(a) Long Short-Term Memory (b) Gated Recurrent Unit

Fuente: Chung et al (2014).

2.2.6.5.1. LSTM

Como solución a las deficiencias de las RNN normales, a Hochreiter y Schmidhuber se les ocurrió las redes LSTM, una arquitectura especial de celdas de memoria a largo y corto plazo, se hace más fácil almacenar información durante un largo periodo. Este tipo de redes fue modificado por muchas personas desde entonces, la

formulación de una cela simple de LSTM como se muestra en la Figura 18, también se pueden representar mediante las siguientes ecuaciones (Shewalkar et al., 2019):

$$f_t = \sigma\big(W_f \cdot [h_{t-1}, x_t] + b_f\big) \tag{21}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_f) \tag{22}$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{23}$$

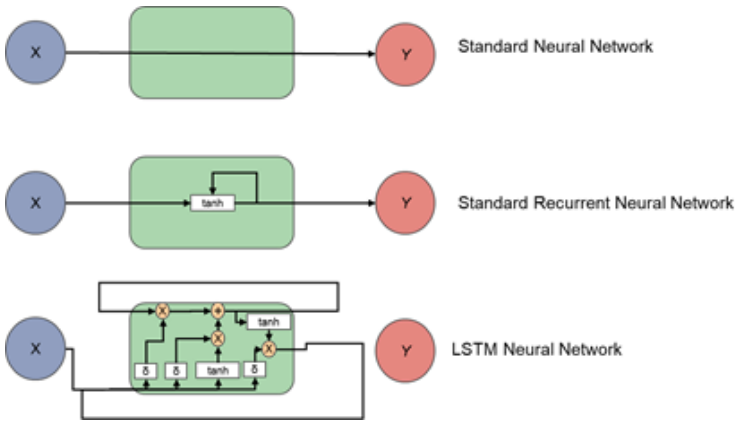
$$C_t = f_t \cdot C_{t-1} + i_f * \check{C}_t) \tag{24}$$

$$o_t = \sigma\big(W_0 \cdot [h_{t-1}, x_t] + b_f\big) \tag{25}$$

$$h_t = o_t * \tanh\,(C_t) \tag{26}$$

Donde x es la entrada, h es la secuencia del vector oculto, W representa la matriz de pesos, b representa el sesgo, σ es la función sigmoide, $tanh$ es la función hiperbólica tangente, finalmente $i, f, o, C, \tilde{C}$, son compuerta de entrada, compuerta de olvido, contenido de la celda de memoria, nuevo contenido de la celda de memoria respectivamente.

Figura 18: Representación de una red neuronal, RNN estándar y una red LSTM.



Fuente: Garzon (2018).

2.2.6.5.2. GRU

De acuerdo a Chung et al. (2014) este tipo de unidad recurrente (Gated Recurrent Unit) fue propuesta para capture de manera adaptada dependencias de diferentes escalas de tiempo, de manera similar a una unidad de LSTM, las unidades GRU tienen unidades de composición que modulan el flujo de información dentro de la unidad, sin tener células de memoria separadas.

2.2.6.6. Métricas de rendimiento

La evaluación de desempeño es un problema importante en la investigación interdisciplinaria, las métricas de rendimiento (medidas de error) son componentes vitales en el marco de la evaluación en varios campos. En los experimentos de regresión en Machine Learning, las métricas de rendimiento se utilizan para comparar las predicciones del modelo entrenado versus las predicciones actuales, de los datos de prueba. El pronóstico tiene una larga historia de empleo de métricas de rendimiento para medir cuanto se desvían las predicciones de las observaciones reales para evaluar la calidad y elegir correctamente los métodos de predicción (Botchkarev, 2018).

Algunas métricas son más populares que las otras, varios investigadores realizaron encuestas de organizaciones y profesionales para comprender la frecuencia de uso o la importancia de las diferentes métricas. Se identificaron una variedad de métricas en estas encuestas. (Botchkarev, 2019). Sin embargo, las mejores métricas y más comunes se muestran en la siguiente Tabla 1.

Tabla 1: Top 3 métricas identificadas.

Métrica	C&A, 1982	M&K, 1995	M et al, 2006	F&G, 2007
Error cuadrático medio (MSE, RMSE)	34	10	6	9
Error absoluto medio (MAE)	18	25	20	36
Error porcentual absoluto medio (MAPE)	15	52	45	44

Fuente: Botchkarev (2019).

De acuerdo a la tabla, en los años 80 se prefería los métodos MSE o RMSE, por otro lado por los años 2000 en adelante MAPE paso adelante con un 44%,y MAE en segundo lugar.

Por otro lado, Chicco et al. (2021) indica como métricas: MAE, MSE, RMSE y MAPE. Estas métricas abarcan toda la rama positiva de la línea real con un límite inferior cero, que implica un ajuste perfecto y crece infinitamente creciendo para empeorar los modelos. Por definición estos valores dependen en gran medida de los rangos de las variables que describen. Matemáticamente los métodos se definen de la siguiente manera:

2.2.6.6.1. Error cuadrático medio (MSE)

Se puede usar MSE si hay valores atípicos que deben detectarse, de hecho, es mejor MSE para atribuir pesos más grandes a dichos puntos.

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(X_i - Y_i)^2 \tag{27}$$

Mejor valor =0, peor valor = +∞

Donde X_i es el valor pronosticado, Y_i es el valor actual y m es la cantidad de datos.

2.2.6.6.2. El error cuadrático medio de la raíz (RMSE)

Las dos cantidades MSE y RMSE están relacionadas a través de la raíz cuadrada

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(X_i - Y_i)^2} \quad (28)$$

Mejor valor =0, peor valor = +∞

Donde X_i es el valor pronosticado, Y_i es el valor actual y m es la cantidad de datos.

2.2.6.6.3. Error absoluto medio (MAE)

MAE se puede utilizar si los valores atípicos representan partes corruptas de los datos, de echo MAE no está penalizando los valores atípicos del entrenamiento, por lo tanto MAE proporciona una medida de rendimiento genérica y limitada para el modelo, por otro lado, si el conjunto de pruebas tiene muchos valores atípicos el rendimiento del modelo será mediocre.

$$MAE = \frac{1}{m}\sum_{i=1}^{m}|X_i - Y_i| \quad (29)$$

Mejor valor =0, peor valor = +∞

Donde X_i es el valor pronosticado, Y_i es el valor actual y m es la cantidad de datos.

2.2.6.6.4. Error porcentual absoluto medio (MAPE)

El error porcentual absoluto medio es otra métrica de rendimiento para los modelos de regresión, teniendo una interpretación muy intuitiva en términos de error relativo: debido a su definición, se recomienda su uso en las tareas en las que es más importante que sea sensible a las variaciones relativas que a las variaciones absolutas. Sin embargo, también tiene varios inconvenientes, los más críticos son la restricción a su uso a datos estrictamente positivos por definición.

$$MAPE = \frac{1}{m}\sum_{i=1}^{m}\left|\frac{X_i - Y_i}{Y_i}\right| \quad (30)$$

Mejor valor =0, peor valor = +∞

Donde X_i es el valor pronosticado, Y_i es el valor actual y m es la cantidad de datos.

2.3. HIPÓTESIS DE LA INVESTIGACIÓN

Un modelo basado en Machine Learning optimizará el pronóstico de ventas de la empresa Ricos Pan, año 2020 - 2021.

2.4. VARIABLES

2.4.1. Variable dependiente

Modelo de pronóstico.

2.4.2. Variable independiente

Rendimiento del pronóstico.

2.5. OPERACIONALIZACIÓN DE VARIABLES

VARIABLE	Dimensión, indicador	Valor
Modelo de pronóstico	Método de pronóstico	Series de tiempo • Redes neuronales convolucionales (CNN) • Redes neuronales recurrentes (LSTM) • Redes neuronales recurrentes (GRU) • Máquina de soporte vectorial de regresión (SVR) Métodos de juicio • Manual o de juicio
Rendimiento del pronostico	Medición del desempeño - Error porcentual absoluto medio (MAPE)	Mejor valor =0, peor valor = $+\infty$

CAPÍTULO III

MATERIALES Y MÉTODOS

3.1. TIPO Y DISEÑO DE LA INVESTIGACIÓN

De acuerdo a Hernández Sampieri et al. (2014) la ruta cuantitativa es apropiada cuando queremos estimar las magnitudes u ocurrencias de los fenómenos y probar hipótesis. Para la presente investigación el enfoque cuantitativo resulta ser más apropiado.

3.1.1. Tipo de investigación

Naghi Namakforoosh (2005) señala que el experimento se puede definir como una investigación científica en la cual el investigador manipula y controla una o más variables, de acuerdo a este autor la investigación actual seria de tipo experimental.

3.1.2. Diseño de la investigación

En los diseños cuasi experimentales, los sujetos no se asignan al azar los grupos ni se emparejan, si no dichos grupos ya están formados antes del experimento: son grupos intactos y la manera en que se formaron es independiente del experimento (Hernández Sampieri et al., 2014). La presente investigación será cuasiexperimental ya que la muestra no será de forma aleatoria.

3.2. POBLACIÓN Y MUESTRA

3.2.1. Población

La población está representada por los 76 productos que se venden en el área de Panadería de la empresa Ricos Pan.

3.2.2. Muestra

De acuerdo a Hernández Sampieri et al. (2014), las muestras pueden ser divididas en dos grandes ramas, probabilísticas y no probabilísticas. Para la presente investigación se adoptó una muestra no probabilística, se seleccionó los 15 productos más vendidos durante los años 2020 y 2021 al tener mayor rotación y variación dentro de la empresa.

3.3. MÉTODO DE RECOLECCIÓN DE DATOS

Los datos se obtuvieron de manera directa, a través del sistema de control interno que maneja Ricos Pan.

3.4. MÉTODOS DE TRATAMIENTO DE DATOS

Para el tratamiento de datos se utilizó el lenguaje de Python, los datos los proporciono la empresa a través del sistema del control interno en formato CSV y JSON, en general para todo el manejo de datos se utilizó una clase llamada `ProductoDatos`, en donde se definieron diversas funciones y variables necesarias para el análisis de datos, el código de la definición de la clase y variables se muestra a continuación, las principales variables para el tratamiento de datos se muestran en la Tabla 2.

```
import os # Librería para leer archivos del disco
import numpy as np # Utilizado para operaciones
import pandas as pd # Librería para analizar datos
import json # Librería para trabajar con el formato JSON
import requests
class ProductoDatos:
    producto_id = 0
    nombre = ""
    datos = pd.DataFrame()
    predicciones_entrenamiento=[]
    predicciones_prueba=[]
    precios = []
    quantil = 0
    mape = 0
    smape = 0
    mse = 0
    mae = 0
    fecha_inicial = ""
    fecha_final = ""
```

Tabla 2: Definición de variables de la clase ProductoDatos.

Variable	Tipo	Observación
`producto_id`	Numérico	Es un número que identifica al producto
`nombre`	Cadena	El nombre del producto
`datos`	Dataframe	Esta variable contiene los datos de venta del producto
`quantil`	Numérico	Se usó en el análisis de valores atípicos (outliers).
`Mape, smape, mse, mae`	Numérico	Es donde se almacena la taza de errores de las predicciones
`fecha_inicial, fecha_final`	Fecha	Es donde se almacena la fecha inicial y final del periodo de ventas
`predicciones_entrenamiento, predicciones_prueba`	Array	Se usa para almacenar las predicciones de entrenamiento y prueba

Elaboración propia.

3.4.1. Carga de datos

Los datos los proporciono el sistema de control interno de la empresa Ricos Pan, cuyo formato es CSV, y cuya estructura consta de la cantidad, la fecha y hora de la venta, y el precio separados por comas, como se muestra en la Figura 19.

Figura 19: Ejemplo de datos de venta

```
1  cantidad,fecha,precio
2  4,"2017-01-15 15:23:56",0.25
3  8,"2017-01-15 16:56:05",0.25
4  8,"2017-01-15 16:58:29",0.25
5  4,"2017-01-15 17:49:29",0.25
6  4,"2017-01-15 18:10:24",0.25
7  4,"2017-01-15 18:15:45",0.25
8  2,"2017-01-15 18:17:17",0.25
9  4,"2017-01-15 18:18:42",0.25
10 4,"2017-01-15 18:20:40",0.25
11 6,"2017-01-15 18:27:19",0.25
12 6,"2017-01-15 18:28:56",0.25
```

Elaboración propia.

Para la carga de datos se usó la función `read_csv` de la librería pandas, que se encargar de leer archivos CSV, los archivos se encuentran en la carpeta `data` en la raíz del proyecto, así mismo, se definió la fecha como índice del dataframe.

```
def cargarDatos(self):
    file = 'data/' + str(self.producto_id) + '.csv'
    self.datos = pd.read_csv(file)
    self.datos['fecha'] = pd.to_datetime(self.datos['fecha'])
    self.datos = self.datos[["fecha", "cantidad", "precio"]]
    self.datos = self.datos.set_index('fecha')
```

3.4.2. Exploración de datos

Para la exploración de datos se utilizó la tabla de frecuencias y gráficos generados por la librería `pandas` y `matplotlib` de Python, también se utilizó la clase `ProductosDatos` para la carga de datos. Para explicar de mejor manera se utilizará como ejemplo la venta del producto PAN CIABATTA, primero cargamos los datos utilizando la función `cargarDatos` de la clase `ProductoDatos`, definida y explicada en el punto anterior.

```
import numpy as np
import pandas as pd
from tabulate import tabulate
import matplotlib.pyplot as plt
from data import ProductoDatos
pd.options.display.float_format = '${:,.2f}'.format
producto=ProductoDatos(3366,'PVP. PAN CIABATTA')
producto.cargarDatos()
```

Generamos la tabla de frecuencias mediante la función `value_counts` que devuelve una serie que contiene los distintos valores incluidos junto con el número de apariciones de cada uno como se muestra en la Tabla 3, el código para se muestra a continuación.

```
frecuencias = pd.DataFrame(producto.datos["cantidad"].value_counts())
frecuencias = frecuencias.rename(columns={'cantidad':'numero_veces'})
frecuencias["cantidad"]=frecuencias.index
frecuencias = frecuencias[["cantidad","numero_veces"]]

frecuencias.sort_index(inplace=True)
frecuencias["cantidad_acumulada"] = frecuencias["cantidad"].cumsum()
```

```
frecuencias["porcentaje"] = ((frecuencias["cantidad"] /
          frecuencias["cantidad"].sum()).round(4))*100
frecuencias["porcentaje_acumulado"] = ((frecuencias["cantidad"].cumsum()/
                          frecuencias["cantidad"].sum()).round(4))*100

print(tabulate(frecuencias, headers = 'keys', tablefmt = 'psql'))
```

Tabla 3: Tabla de frecuencias del producto pan ciabatta.

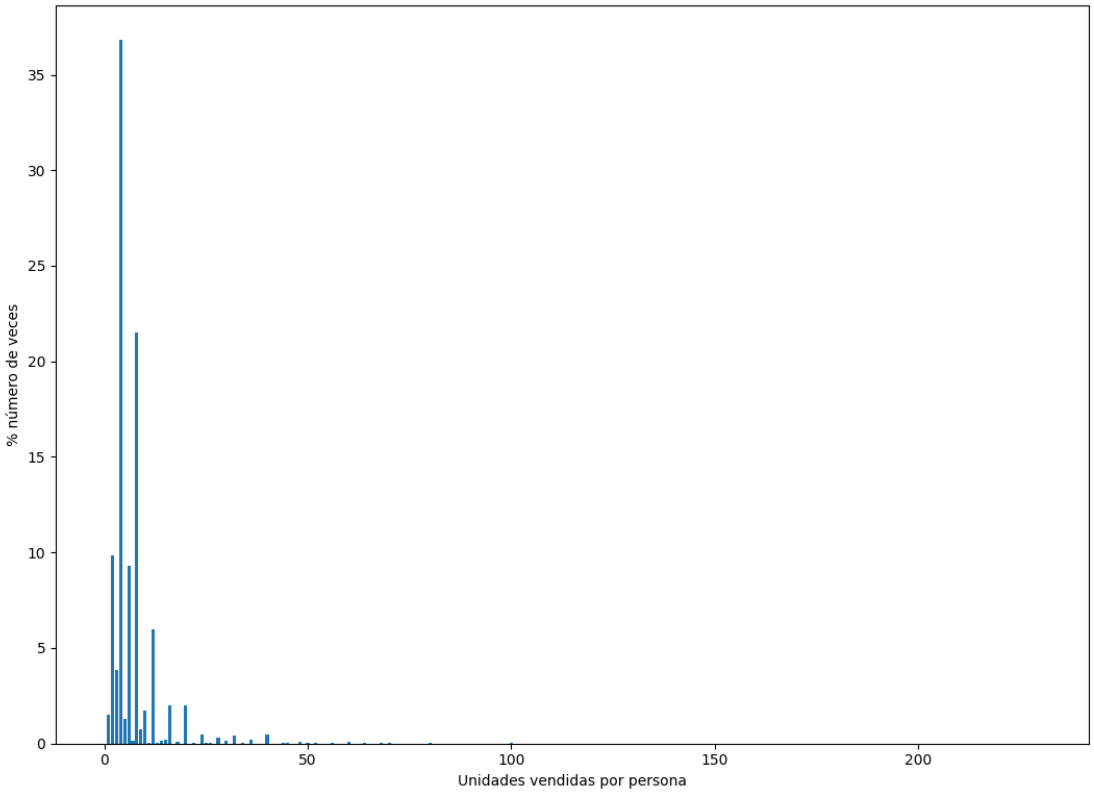
Unidades vendidas por persona	Número veces	Cantidad Acumulada	Porcentaje (%)	Porcentaje acumulado
0	2	2	0	0
1	1721	1723	1.53	1.53
2	11090	12813	9.83	11.35
3	4343	17156	3.85	15.2
4	41542	58698	36.81	52.02
5	1434	60132	1.27	53.29
6	10507	70639	9.31	62.6
7	194	70833	0.17	62.77
8	24292	95125	21.53	84.3
9	857	95982	0.76	85.06
10	1915	97897	1.7	86.75
11	33	97930	0.03	86.78
12	6754	104684	5.99	92.77
13	19	104703	0.02	92.78
14	155	104858	0.14	92.92
15	242	105100	0.21	93.14
16	2239	107339	1.98	95.12
17	5	107344	0	95.12
18	108	107452	0.1	95.22
19	9	107461	0.01	95.23
20	2256	109717	2	97.23
21	15	109732	0.01	97.24
22	23	109755	0.02	97.26
23	6	109761	0.01	97.27
24	538	110299	0.48	97.74
25	37	110336	0.03	97.78
26	42	110378	0.04	97.81
27	10	110388	0.01	97.82
28	365	110753	0.32	98.15
29	1	110754	0	98.15
30	153	110907	0.14	98.28

Elaboración propia.

De acuerdo a la Tabla 3 observamos que las personas tienden a comprar 4 y 8 unidades en su mayoría, que representan el 36% y 21% respectivamente, por otro lado, podemos observar que el 92% por ciento de personas compran entre 1 y 12 panes. Para tener una mejor perspectiva graficamos la cantidad vs % de número de veces.

```
plt.bar(frecuencias.index,frecuencias['porcentaje'])
plt.xlabel("cantidad")
plt.ylabel("% número de veces")
plt.show()
```

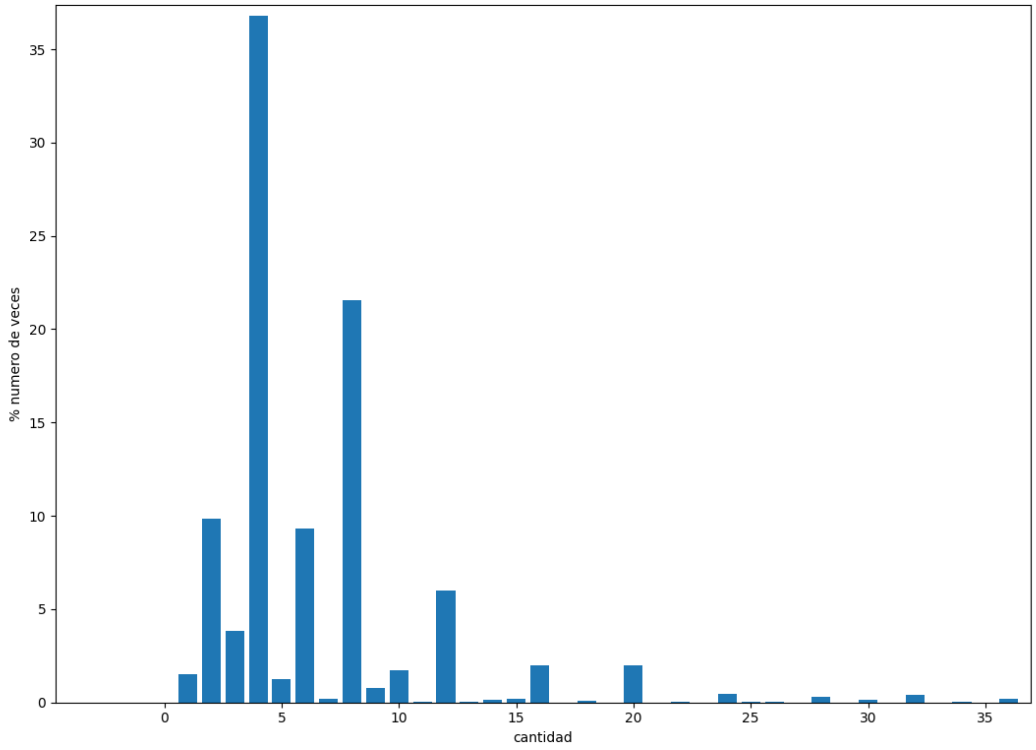
Figura 20: Unidades vendidas por persona vs % de número de veces de venta.



Elaboración propia.

Como se puede observar la Figura 20 las compras fueron desde 1 unidad hasta 230 unidades por persona, para poder analizar mejor ampliamos en el rango de 1 a 35 unidades.

Figura 21: Unidades vendidas por persona vs % de número de veces ampliado.



Elaboración propia.

En la Figura 21 se puede observar el rango de cantidad de compra de panes que esta entre 1 y 12 unidades que representa el 92% de ventas, el porcentaje restante se pueden considerar como valores atípicos.

3.4.3. Preprocesamiento de datos e ingeniera de características

Para el preprocesamiento de datos se consideraron eliminar los valores atípicos haciendo uso del diagrama y cajas y bigotes, teniendo en cuenta los rangos intercuantiles, por otro lado, también se consideró borrar las ventas en 0 ya que son datos errados, debido a que no pueden existir ventas de productos en 0. Luego del preprocesamiento de datos se prosiguió con la ingeniería de características que consiste en agregar ciertas características al conjunto te datos como por ejemplo si el día de la venta fue un fin de semana, o fue algún día de alguna festividad como lo es el día de la madre, en donde tiene un gran impacto la venta de tortas, y de esta manera ayudar a que el modelo nos de mejores pronósticos.

3.4.3.1. Valores en cero

De acuerdo a la Tabla 3 existen filas que contienen 0, se borraron mediante la función `borrarCeros` de la clase `productoDatos`.

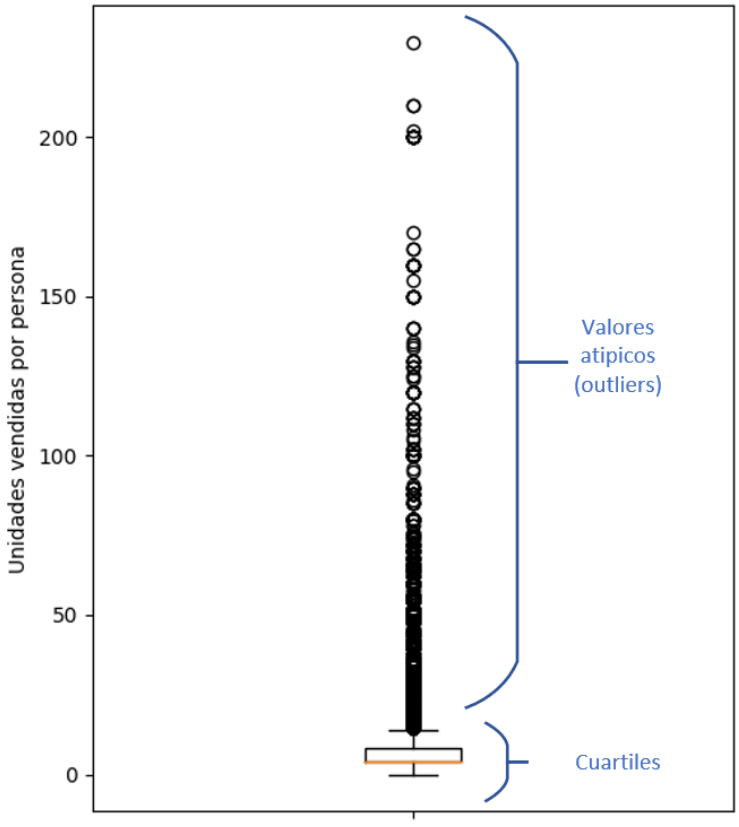
```
def borrarCeros(self):
    self.datos=self.datos.loc[(self.datos['cantidad'] != 0)]
```

3.4.3.2. Valores atípicos (outliers)

Para analizar los valores atípicos se usó el diagrama de cajas y bigotes. Se utilizo el siguiente código para poder graficar y así visualizar de mejor manera la distribución de datos:

```
fig = plt.figure(figsize=(10, 7))
plt.boxplot(producto.datos['cantidad'])
plt.show()
```

Figura 22: Grafico de cajas y bigotes del producto pan ciabatta.



Elaboración propia.

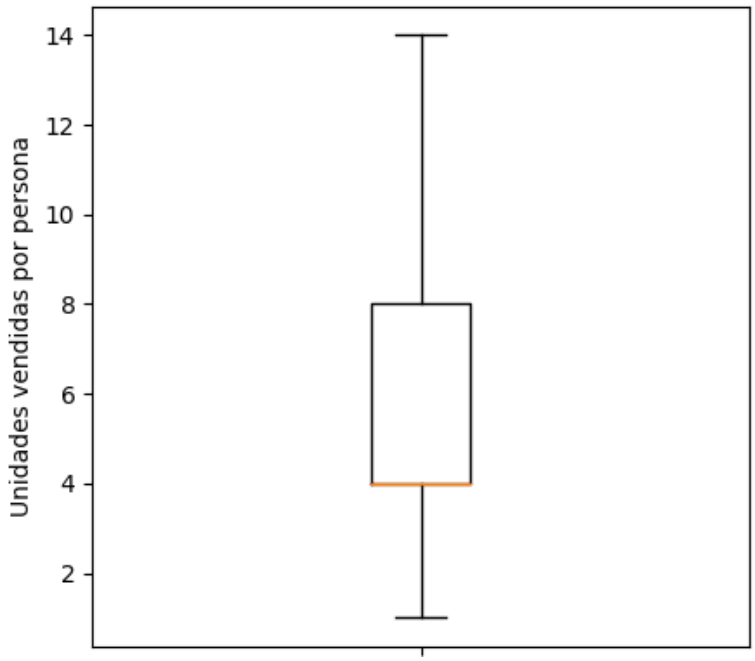
Como se observa en la Figura 22 se muestra las unidades vendidas por persona del pan ciabatta, así mismo los valores atípicos están representados por ciruclos que van

desde 14 a más de 250, por lo que se optó por eliminarlos, considerando a aquellos que están por lo menos 1.5 veces el rango intercuantil (Q3-Q1) del borde de la caja, solo se consideró la parte superior de la caja, para ello se implementó la siguiente función:

```
def borrarValoresAtipicos(self):
    q3 = self.datos["cantidad"].quantile(0.75)
    q2 = self.datos["cantidad"].quantile(0.25)
    riq = q3-q2
    max = q3+1.5*riq
    self.datos = self.datos.drop(self.datos[self.datos.cantidad >max].index)
```

La figura 23 muestra los valores después de borrar los valores atípicos.

Figura 23: Cajas y bigotes del producto pan ciabatta sin los valores atípicos.



Elaboración propia.

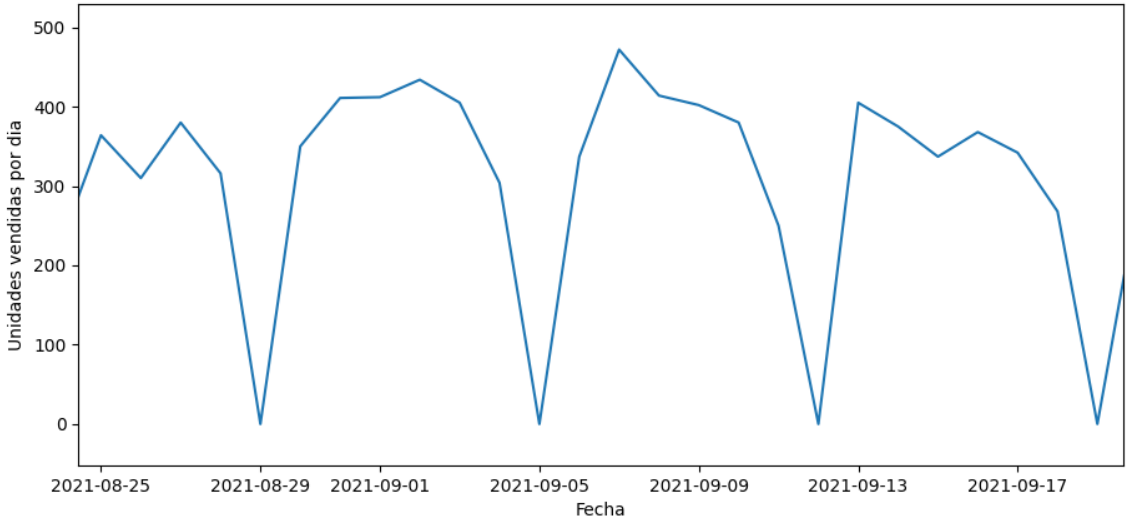
De acuerdo a la Figura 23 se observa que los valores atípicos fueron eliminados y solo se conservó los valores típicos que van desde 1 a 14, siendo en su mayoría las ventas de entre 2 y 8 unidades de panes ciabatta por persona.

3.4.3.3. Ingeniería de características

Características generales

Como se mencionó en el marco teórico, en esta etapa se crearán características adicionales con los datos para poder ayudar a los modelos de Machine Learning a producir mejores resultados, para lo cual realizamos un análisis visual de algunos productos.

Figura 24: Ventas por día del producto Pan ciabatta.



Elaboración propia

Como se puede observar en la Figura 24 existen días en que la venta es 0, exactamente todos los días domingo, es por ellos que creamos una función que agregue un indicador de días de la semana, en este caos 1 es lunes y 2 para martes, así sucesivamente hasta domingo que es representado por 7, para así poder proporcionar más datos de entrada a nuestros modelos de basados en Machine Learning, las características generales que se agregaron se listan a continuación:

- **Dia:** número del 1 al 31 que representa el día del mes
- **Dia semana:** un número del 0 al 6 que representan los días de la semana (lunes a domingo).

- **Mes:** número del 0 al 11 que representan los meses del año.
- **Semana:** Numero del 0 al 51 que representan la semana del año.
- **Año:** Numero del año

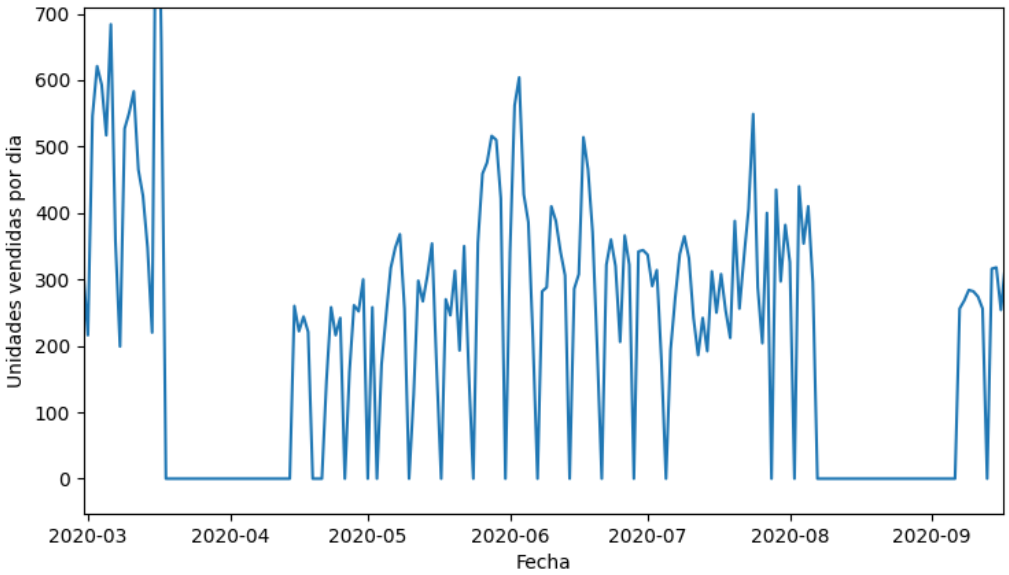
Para agregar las características, se implementó la función `agregarCaracteristicasGenerales` dentro de la clase `ProductoDatos`, usamos las funciones *`day`*, *`dayofweek`*, *`month`*, *`week`* y *`year`* que trae pandas.

```
def agregarCaracteristicasGenerales(self):
    precios = pd.DataFrame.copy(self.datos.resample('D').mean())
    self.datos = self.datos.resample('D').sum()
    self.datos['precio'] = precios['precio']
    self.datos = self.datos.sort_index()
    idx = self.datos.index
    idx_fecha_inicio = idx[0]
    idx_fecha_final = idx[-1]
    indice_completo = pd.date_range(idx_fecha_inicio, idx_fecha_final)
    self.datos = self.datos.reindex(indice_completo)
    self.datos = self.datos.fillna(0)
    self.datos['dia'] = self.datos.index.day
    self.datos['dia_semana'] = self.datos.index.dayofweek
    self.datos['mes'] = self.datos.index.month
    self.datos['semana'] = pd.Int64Index(
                            self.datos.index.isocalendar().week)
    self.datos['year'] = self.datos.index.year
```

Local cerrado

Por otro lado, si analizamos la venta histórica del año 2020 de acuerdo a la Figura 25, existen dos intervalos de fecha en que las ventas fueron 0, esto fue debido a que el local cerro por pandemia de coronavirus, las fechas exactas fueron del 23/03/2020 al 2020/04/015 y del 07/08/2020 al 07/09/2020.

Figura 25: Ventas por día del producto en el año 2020.



Elaboración propia.

Para dicha característica creamos la función `localCerrado`, en donde especificamos el rango de fechas del local cerrado, nos apoyamos con la librería `pandas` para poder agregar una columna llamada `local_cerrado`, en donde 0 significa que está abierto y 1 significa que está cerrado el código es el siguiente:

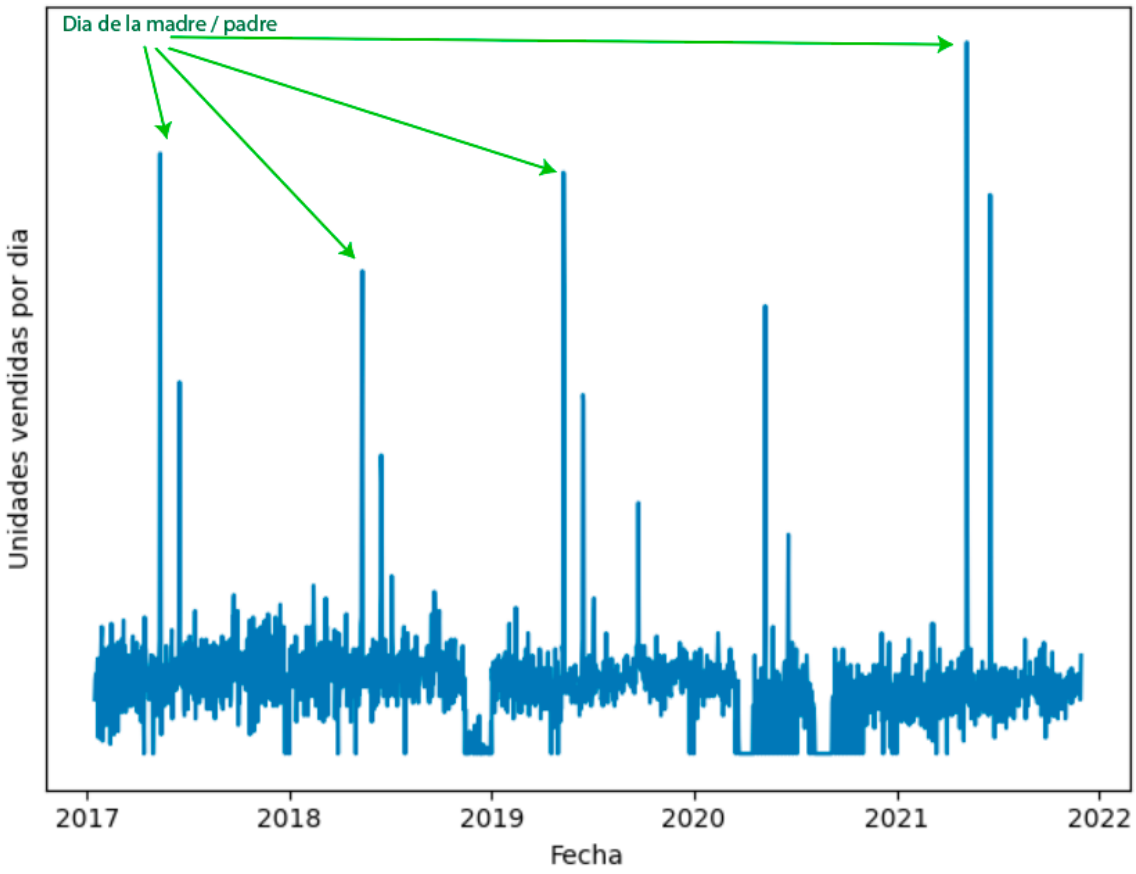
```
def localCerrado(self):
    self.datos['local_cerrado'] = np.zeros(len(self.datos.index))
    fecha_inicial = pd.Timestamp("2020-03-23")
    fecha_final = pd.Timestamp("2020-04-15")
    index = self.datos.index
    idx_bools = ((index >= fecha_inicial) & (index <= fecha_final))
    self.datos.loc[idx_bools, 'local_cerrado'] = 1
    fecha_inicial = pd.Timestamp("2020-08-07")
    fecha_final = pd.Timestamp("2020-09-07")
    index = self.datos.index
    idx_bools = ((index >= fecha_inicial) & (index <= fecha_final))
    self.datos.loc[idx_bools, 'local_cerrado'] = 1
```

Días festivos

Si analizamos los datos de la venta de tortas notaremos que existen picos de venta, corresponden a días festivos, día de la madre y día del padre, que ocurren cada primer domingo de mayo y segundo domingo de junio respectivamente como se puede observar en la Figura 26. Las ventas no ocurren exactamente los días centrales, ocurren durante

días antes, siendo el pico de venta un día antes del día central. Esto solo afectar a las tortas, mas no a los demás productos o afectan en menor medida.

Figura 26: Venta histórica de torta clásica mediana.



Elaboración propia.

Para dicha característica creamos la función llamada `diasFestivos`, en donde definimos un diccionario de fechas y le damos un peso a cada fecha, como se explicó las ventas no ocurren el día central si no, días antes.

```
def diasFestivos(self):
    self.datos['dias_festivos'] = np.zeros(len(self.datos))
    dias_festivos = {
        # dia de la madre
        "2017-05-12": 0.5,
        "2017-05-13": 1,
        "2018-05-11": 0.5,
        "2018-05-12": 1,
        "2019-05-10": 0.5,
        "2019-05-11": 1,
        "2020-05-08": 0.5,
        "2020-05-09": 1,
        "2021-05-07": 0.5,
        "2021-05-08": 1,
```

```
        # dia padre
        "2017-06-16": 0.5,
        "2017-06-17": 1,
        "2017-06-18": 0.5,
        "2018-06-15": 0.5,
        "2018-06-16": 1,
        "2018-06-17": 0.5,
        "2019-06-14": 0.5,
        "2019-06-15": 1,
        "2021-06-18": 0.5,
        "2021-06-19": 1,
        "2021-06-20": 0.5,
    }
    tortas=[3323,3296,3397,3299,3416,3417,3421,3425,3426,3428]
    if self.producto_id in tortas:
        for key, value in dias_festivos.items():
            self.datos.at[key, "dias_festivos"] = value
```

El código que agrupa el preprocesamiento de datos y agrega características, se muestra a continuación, cabe resaltar que para el precio la agregación de datos no es por suma si no por promedio, el ejemplo se muestra para una torta de 26 cm, el resultado se muestra en la Tabla 4.

```
import pandas as pd
from tabulate import tabulate
import matplotlib.pyplot as plt
from data import ProductoDatos
pd.options.display.float_format = '{:,.2f}'.format
producto=ProductoDatos(3299,'TORTA 26 CM')
producto.cargarDatos()
producto.borrarCeros()
producto.borrarValoresAtipicos()

precios = pd.DataFrame.copy(producto.datos.resample('D').mean())
producto.datos = producto.datos.resample('D').sum()
producto.datos['precio'] = precios['precio']
producto.agregarCaracteristicasGenerales()
producto.diasFestivos()
print(producto.datos.head(-10))
```

Tabla 4: Datos incluyendo características

Fecha	Cantidad	Precio	Dia	Dia semana	Mes	Semana	Año (Year)	Días festivos
15/01/2017	17	38	15	6	1	2	2017	0
16/01/2017	23	38	16	0	1	3	2017	0
17/01/2017	25	38	17	1	1	3	2017	0
18/01/2017	25	38	18	2	1	3	2017	0
19/01/2017	22	38	19	3	1	3	2017	0
...	...	...	...	...	...	...	...	...
16/11/2021	25	42	16	1	11	46	2021	0
17/11/2021	24	42	17	2	11	46	2021	0
18/11/2021	18	42	18	3	11	46	2021	0
19/11/2021	23	42	19	4	11	46	2021	0
20/11/2021	23	42	20	5	11	46	2021	0

Elaboración propia.

Para poder tener toda la muestra ya que son 15 productos, se implementó la función `cargarTransformarDatos` que se encarga de cargar los datos y transformarlos, este retorna una lista de productos, en donde están todos los datos necesarios para la siguiente etapa que es el modelado y entrenamiento.

```
def cargarTransformarDatos():
    lista_productos = []
    with open("data/lista_productos.json", 'r') as archivo:
        lista_productos_json = json.load(archivo)

    for producto in lista_productos_json:
        obj_producto = ProductoDatos(producto['producto_id'],
producto['nombre'])
        obj_producto.precios = producto['precios']
        obj_producto.cargarDatos()
        obj_producto.borrarCeros()
        # Datos no agrupados
        obj_producto.borrarValoresAtipicos()
        # Agrupar datos y agregar caracteristicas
        obj_producto.agregarCaracteristicasGenerales()
        obj_producto.localCerrado()
        obj_producto.diasFestivos()
        lista_productos.append(obj_producto)
    return lista_productos
```

3.4.4. Construcción e implementación de modelos Machine Learning

Luego de haber realizado las diferentes técnicas de preprocesamiento de los datos, se procedió a implementar el entrenamiento y prueba de los modelos de Machine

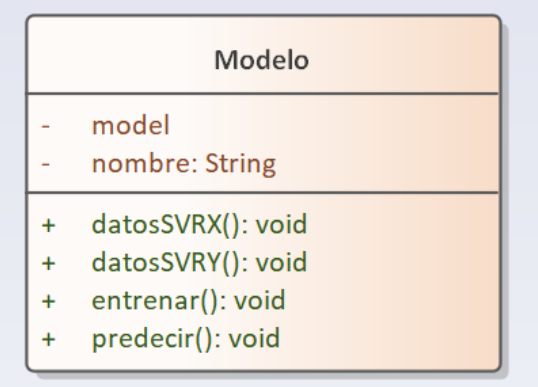
Learning, los modelos implementados fueron: SVR, LSTM, GRU y CONV. Para implementar SVR se utilizó la librería `scikit learn` a diferencia de los modelos LSTM, GRU y CONV se utilizó `Tensor Flow` y `Keras`.

- SVR: Modelo basado en una máquina de soporte vectorial para regresión.
- LSTM: Modelo basado en redes neuronales recurrentes.
- GRU: Modelo basado en redes neuronales recurrentes.
- CONV: Modelo basado en redes neuronales convolucionales.

Modelos

Para trabajar con los modelos se implementó una clase simple llamada *Modelo*, la principal razón de la implementación fue que SVR no admite los mismos tipos de entrada que los modelos LSTM, GRU y CONV, por lo tanto, para poder interactuar con los 4 modelos a la vez, los métodos y atributos se muestran en la Figura 27.

Figura 27: Clase modelo a implementar.



Elaboración propia.

- **Atributos:**
  - **Model:** Este atributo contiene el modelo
  - **Nombre:** Atributo que indica el nombre del modelo

- **Operaciones:**
  - **Entrenar:** Se usa para para entrenar el modelo, como parámetros de entrada tuvo: `dataX` (datos de entrada), `dataY` (datos de salida), `batch_size` (tamaño del batch de entrenamiento), `epochs` (número épocas de entrenamiento), `validation_data` (datos de prueba), `callbacks` (llamadas durante el entrenamiento), `verbose` (si se muestra detalladamente el entrenamiento). No devuelve ningún dato, entrena el modelo, y guarda los modelos para que en la etapa de despliegue pueda ser consumido.
    - **Predecir:** Se usa para predecir valores
    - **DatosSVRX:** Se uso para transformar los valores de entrada del modelo SVR
    - **DatosSVRY:** Se utilizo para transformar los valores de salida del modelo SVR

La implementación del modelo fue la siguiente:

```
import numpy as np
import tensorflow as tf
class Modelo:
    def __init__(self, model, nombre):
        self.model = model
        self.nombre = nombre

    def entrenar(self,
                 dataX,
                 dataY,
                 batch_size,
                 epochs,
                 validation_data,
                 callbacks,
                 verbose):
        if self.nombre == "SVR":
            Xtrain=self.datosSVRX(dataX)
            Ytrain=self.datosSVRY(dataY)
            self.model.fit(Xtrain,Ytrain)
        else:
            self.model.fit(
                dataX,
                dataY,
                batch_size=batch_size,
                epochs=epochs,
                validation_data=validation_data,
                callbacks=callbacks,
```

```
                verbose=verbose,
            )

    def predecir(self,data,mejor_modelo_nombre=""):
        if self.nombre == "SVR":
            return self.model.predict(self.datosSVRX(data)).reshape(-1, 1)
        else:
            mejor_modelo = tf.keras.models.load_model(mejor_modelo_nombre)
            return mejor_modelo.predict(data).reshape(-1, 1)

    def datosSVRX(self, dataX):
        X = []
        T=dataX.shape[1]
        for data in dataX:
            first=data[:,0:1].reshape(1,-1)
            X.append(first)
        return np.array(X).reshape(-1,T)

    def datosSVRY(self,dataY):
        Y = []
        for data in dataY:
            Y.append(data)
        return np.array(Y).ravel()
```

Luego de la implementación de la clase definimos los modelos a usar:

```
LATENT_DIM = 6
T = 6
BATCH_SIZE = 32
HORIZON = 1
modelos = []

model1 = Sequential()
model1.add(GRU(LATENT_DIM, input_shape=(T, len(columnas_validas))))
model1.add(Dense(HORIZON))
model1.compile(optimizer="RMSprop", loss="mse", metrics=['mse'])
modelos.append(Modelo(model1, "GRU"))

model2 = tf.keras.models.Sequential([
    tf.keras.layers.LSTM(LATENT_DIM, return_sequences=False),
    tf.keras.layers.Dense(units=1)
])
model2.compile(optimizer="RMSprop", loss="mse", metrics=['mse'])
modelos.append(Modelo(model2, "LSTM"))

model3 = tf.keras.Sequential([
    tf.keras.layers.Conv1D(filters=32,
                           kernel_size=(T,),
                           activation='relu'
                           ),
    tf.keras.layers.Dense(units=32, activation='relu'),
    tf.keras.layers.Dense(units=1),
])
model3.compile(optimizer="RMSprop", loss="mse", metrics=['mse'])
modelos.append(Modelo(model3, "Conv1D"))

model4 = SVR()
modelos.append(Modelo(model4, "SVR"))
```

En las Figuras 28,29,30 y 31 se muestra el resumen de cada modelo, los modelos se definen a través de `Sequential` que sirve para agrupar capas en forma de pila a través de `tf.keras.Model`.

Figura 28: Resumen del modelo GRU.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
gru (GRU)                    (None, 6)                 306
_________________________________________________________________
dense (Dense)                (None, 1)                 7
=================================================================
Total params: 313
Trainable params: 313
Non-trainable params: 0
```

Elaboración propia

En la Figura 28 se puede observar dos capas la primera `GRU` con 6 unidades y 1 capa denominada `dense`, que se refiere a una capa regular de conexión profunda.

Figura 29: Resumen del modelo LSTM.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 6)                 384
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 7
=================================================================
Total params: 391
Trainable params: 391
Non-trainable params: 0
```

Elaboración propia.

En la Figura 29 se puede observar dos capas la primera `LSTM` con 6 unidades y 1 capa denominada `dense`, que se refiere a una capa regular de conexión profunda.

Figura 30: Resumen del modelo CNN

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 1, 32)             1760
_________________________________________________________________
dense_2 (Dense)              (None, 1, 32)             1056
_________________________________________________________________
dense_3 (Dense)              (None, 1, 1)              33
=================================================================
Total params: 2,849
Trainable params: 2,849
Non-trainable params: 0
```

Elaboración propia.

En la Figura 29 se puede observar 3 capas la primera `Conv1d,` `2` capa denominadas `dense`, que se refiere a capas regular de conexión profunda.

Figura 31: Resumen del modelo SVR

```
{'C': 1.0, 'cache_size': 200, 'coef0': 0.0, 'degree': 3, 'epsilon': 0.1, 'ga
mma': 'scale', 'kernel': 'rbf', 'max_iter': -1, 'shrinking': True, 'tol': 0.
001, 'verbose': False}
```

Elaboración propia.

En la Figura 30 se puede observar los parámetros y el kernel del modelo SVR.

3.4.4.1. Entrenamiento y prueba

Para el entrenamiento y prueba se implementó una función llamada `EntrenarProbar,` cuyos parámetros de entrada fueron los siguientes:

- **Modelo:** Se usa para referirse a un objeto de la clase Modelo en este caso fueron 4 objetos que representan los modelos usados (SVR, LSTM, GRU, CONV).
- **Lista_productos:** Es una lista de objectos de la clase `ProductoDatos,` en donde están definidos los datos de entrenamiento y prueba.

- **Fecha_prueba:** Se usa para dividir los datos de entrenamiento y prueba.
- **Columnas_validas:** Es una lista de las características que tienen nuestros datos.
- **Col_target:** Es el nombre de la columna que se va predecir, en este caso es **cantidad**
- **T:** es el tamaño de la ventada de tiempo, en que los datos se agruparan, en este caso son 6 días y se predice el séptimo.
- **EPOCHS:** La cantidad de épocas que se va entrenar.
- **BATCH_SIZE:** El tamaño del batch, este dato se considera en los parámetros de LSTM, GRU y CONV.
- **Log:** de tipo booleano, que indica si los datos se transformaran con la función logaritmo.
- **Diff:** de tipo booleano, que indica si los datos se van a transformar mediante la función diferencia.

Todas las variables descritas son los parámetros de entrada de la función `EntrenarProbar`

```
def EntrenarProbar(modelo,
                   lista_productos,
                   fecha_prueba,
                   columnas_validas,
                   col_target,
                   T,
                   EPOCHS,
                   BATCH_SIZE,
                   log=False,
                   diff=False):
    resultados = []
    nombre_modelo = modelo.nombre
```

La variable `HORIZON`, controla el horizonte de predicciones en este caso fue de 1 día, `DELTA` se utilizó para la transformación de coordenadas, solo se utilizó la traslación mas no la rotación, debido a la restricción de la función logaritmo en 0.

```
HORIZON = 1
DELTA = 10
```

Se itero todos los productos de la muestra, así mismo se creó la variable `mejor_modelo_nombre` que sirve como nombre del modelo para guardarlo en el disco para luego recuperar el archivo en la etapa de despliegue, en el nombre se indicó si los datos de entrenamiento fueron transformados por la función logaritmo, o la función diferencia, también se indica el código del producto y el número de épocas de entrenamiento.

```
for producto in lista_productos:
    mejor_modelo_nombre = \
        'modelos/best_model.' + nombre_modelo + '.' + \
        ("Log" if log else "noLog") + \
        ("Diff" if diff else "noDiff") + \
        '.' + \
        str(producto.producto_id) +\
        '.' + \
        str(EPOCHS) + '.h5'
```

Dentro del bucle se realizó el escalamiento de los datos, así mismo en caso de que el parámetro log lo indique, se aplicó la función logaritmo y de igual manera con el parámetro `diff`. Cabe resaltar que la función `log` y `diff` solo se aplicó a la columna objetivo, en este caso cantidad. Para las demás características como año, mes, dia_semana, día, días_festivos se utilizó la función `MinMaxScaler` que trae la librería `sklearn`.

```
dff = pd.DataFrame.copy(producto.datos)
dff['cantidad'] = dff['cantidad'] + DELTA  # transformación de coordenadas
scaler_cantidad = MinMaxScaler()
dff['realCantidad'] = dff['cantidad']  # copia de cantidad
scaler_cantidad.fit(dff[[col_target]])
dff['logCantidad'] = np.log(dff['cantidad'])
if log:
    dff[col_target] = np.log(dff[col_target])
else:
    dff[col_target] = scaler_cantidad.transform(dff[[col_target]])

if diff:
    dff['shift' + col_target] = dff[col_target].shift(1)  #
    dff[col_target] = dff[col_target].diff()
    dff = dff.dropna()

# scaler para otras características que no sean la columna objetivo
scaler_caracteristicas = MinMaxScaler()
dff[columnas_validas[1:]] =
scaler_caracteristicas.fit_transform(dff[columnas_validas[1:]])
```

Ventana de datos

Luego del escalado de datos procedemos a construir una ventana de datos, los modelos de esta investigación realizarán las predicciones basadas en una ventana de muestras consecutivas de datos, las características principales de estas ventanas son el ancho (número de pasos de tiempo), el tiempo desplazado entre los pasos, características de entrada (Tensorflow, 2022). Para generar la ventana de datos se utilizó la clase `TimeSeriesTensor` definida por Lazzeri (2020) que realiza los siguientes pasos:

1. Mueve los valores de la serie de tiempo para crear un `DataFrame` que contiene todos los datos y características, necesarias para el entrenamiento.
2. Borra todos los datos que faltan.
3. Transforma el `DataFrame` en tipo `NumPy`, para la entrada de los modelos.

Esta clase toma los siguientes parámetros de entrada

1. `dataset`: serie de tiempo original.
2. `H`: el horizonte de la predicción.
3. `tensor_structure`: es un diccionario para la estructura del tensor de la forma: `{'tensor_name': (range(cambio_máximo_hacia_atrás , cambio_máximo_adelante),[caracteristica, característica, … ])}`
4. `freq`: la frecuencia de la serie de tiempo.
5. `drop_incomplete`: de tipo booleano que indica si se debe borrar los datos incompletos.

La implementación fue de la siguiente manera:

```
# DATOS PARA ENTRENAMIENTO
datos_entrenamiento = dff.copy()[dff.index < fecha_prueba][columnas_validas]
```

```
tensor_structure = {"X": (range(-T + 1, 1), columnas_validas)}
tensor_datos_entrenamiento = TimeSeriesTensor(
    dataset=datos_entrenamiento,
    target=col_target,
    H=HORIZON,
    tensor_structure=tensor_structure,
    freq="D",
    drop_incomplete=True
)
```

El dato de entrada `datos_entrenamiento` tiene el siguiente formato mostrado en la Tabla 5.

Tabla 5: Datos de entrenamiento

	Serie de tiempo escalada			Características adicionales escaladas				
	cantidad	dia_semana	semana	mes	year	dia	local_cerrado	dias_festivos
2021-01-02 00:00:00	0.122807	0.833333	1	0	1	0.0333333	0	0
2021-01-03 00:00:00	0.0614035	1	1	0	1	0.0666667	0	0
2021-01-04 00:00:00	0.166667	0	0	0	1	0.1	0	0
2021-01-05 00:00:00	0.122807	0.166667	0	0	1	0.133333	0	0
2021-01-06 00:00:00	0.0789474	0.333333	0	0	1	0.166667	0	0
2021-01-07 00:00:00	0.140351	0.5	0	0	1	0.2	0	0
2021-01-08 00:00:00	0.114035	0.666667	0	0	1	0.233333	0	0
2021-01-09 00:00:00	0.184211	0.833333	0	0	1	0.266667	0	0
2021-01-10 00:00:00	0.0614035	1	0	0	1	0.3	0	0
2021-01-11 00:00:00	0.0701754	0	0.0192308	0	1	0.333333	0	0
2021-01-12 00:00:00	0.0438596	0.166667	0.0192308	0	1	0.366667	0	0
2021-01-13 00:00:00	0.166667	0.333333	0.0192308	0	1	0.4	0	0
2021-01-14 00:00:00	0.105263	0.5	0.0192308	0	1	0.433333	0	0
2021-01-15 00:00:00	0.0789474	0.666667	0.0192308	0	1	0.466667	0	0
2021-01-16 00:00:00	0.131579	0.833333	0.0192308	0	1	0.5	0	0

Elaboración propia

El retorno de la clase `TimeSeriesTensor` se muestra en la Tabla 6.

Tabla 6: Valores retornados por la clase TimeSeriesTensor

	Valor a predecir	Ventana de datos de la serie de tiempo T=6						Características		
	target/y/t+1	X/cantidad/t-5	X/cantidad/t-4	X/cantidad/t-3	X/cantidad/t-2	X/cantidad/t-1	X/cantidad/t	X/dia_semana/t-5	X/dia_semana/t-4	X/dia_semana/t-3
017-01-20T00:00:00.000000000	0.13158	0.02632	0.10526	0.08772	0.09649	0.13158	0.13158	1.00000	0.00000	0.16667
017-01-21T00:00:00.000000000	0.05263	0.10526	0.08772	0.09649	0.13158	0.13158	0.13158	0.00000	0.16667	0.33333
017-01-22T00:00:00.000000000	0.12281	0.08772	0.09649	0.13158	0.13158	0.13158	0.05263	0.16667	0.33333	0.50000
017-01-23T00:00:00.000000000	0.09649	0.09649	0.13158	0.13158	0.13158	0.05263	0.12281	0.33333	0.50000	0.66667
017-01-24T00:00:00.000000000	0.13158	0.13158	0.13158	0.13158	0.05263	0.12281	0.09649	0.50000	0.66667	0.83333
017-01-25T00:00:00.000000000	0.14035	0.13158	0.13158	0.05263	0.12281	0.09649	0.13158	0.66667	0.83333	1.00000
017-01-26T00:00:00.000000000	0.07895	0.13158	0.05263	0.12281	0.09649	0.13158	0.14035	0.83333	1.00000	0.00000
017-01-27T00:00:00.000000000	0.09649	0.05263	0.12281	0.09649	0.13158	0.14035	0.07895	1.00000	0.00000	0.16667
017-01-28T00:00:00.000000000	0.04386	0.12281	0.09649	0.13158	0.14035	0.07895	0.09649	0.00000	0.16667	0.33333
017-01-29T00:00:00.000000000	0.06140	0.09649	0.13158	0.14035	0.07895	0.09649	0.04386	0.16667	0.33333	0.50000
017-01-30T00:00:00.000000000	0.09649	0.13158	0.14035	0.07895	0.09649	0.04386	0.06140	0.33333	0.50000	0.66667
017-01-31T00:00:00.000000000	0.07895	0.14035	0.07895	0.09649	0.04386	0.06140	0.09649	0.50000	0.66667	0.83333
017-02-01T00:00:00.000000000	0.14912	0.07895	0.09649	0.04386	0.06140	0.09649	0.07895	0.66667	0.83333	1.00000
017-02-02T00:00:00.000000000	0.07018	0.09649	0.04386	0.06140	0.09649	0.07895	0.14912	0.83333	1.00000	0.00000
017-02-03T00:00:00.000000000	0.06140	0.04386	0.06140	0.09649	0.07895	0.14912	0.07018	1.00000	0.00000	0.16667

Elaboración propia

De acuerdo a Tabla 6 la ventana de datos es generada gracias a la clase `TimeSeriesTensor,` el tamaño de la ventana se define por el valor de T.

Entrenamiento y predicción de valores

Lo datos necesarios para el entrenamiento son la serie de tiempo transformada a ventanas de datos, tamaño de lote (batch) y el número de épocas para entrenamiento.

```
modelo.entrenar(
    tensor_datos_entrenamiento["X"],
    tensor_datos_entrenamiento["target"],
    batch_size=BATCH_SIZE,
    epochs=EPOCHS,
    validation_data=(tensor_datos_prueba["X"], tensor_datos_prueba["target"]),
    callbacks=[check_point],
    verbose=1,
)
predicciones_prueba=modelo.predecir(tensor_datos_prueba["X"],mejor_modelo_nombre
)
```

3.4.5. Rendimiento del pronóstico

Después de predecir la serie de tiempo, se deben volver a escalar los valores a su forma original para para esto se utilizó la función `create_evaluation_df` que implementó (Lazzeri, 2020) en donde los parámetros de entrada son los valores de predicción, el tensor de datos generado por la clase `TimeSeriesTensor`, el horizonte, el escalador usado, y el indicador de logaritmo, como resultados nos da un `DataFrame` en donde está contenido los valores pronosticados y los reales en su escala original. Luego aplicamos la función Error porcentual medio absoluto (MAPE) entre los datos pronosticados y los reales, además guardamos los resultados obtenidos en la variable `resultados`. El código implementado es el siguiente:

```
predicciones_prueba=modelo.predecir(tensor_datos_prueba["X"],mejor_modelo_nombre
)

# Funcion inversa de diff
if diff:
    predicciones_prueba_undiff = np.zeros(predicciones_prueba.shape)
    i = 0
    for idx in tensor_datos_prueba.dataframe.index:
        idx = idx + dt.timedelta(days=1)  # es un dia mas
        predicciones_prueba_undiff[i, 0] =
             predicciones_prueba[i][0] + dff.at[idx, "shift" + col_target]
        tensor_datos_prueba["target"][i, 0] =
             tensor_datos_prueba["target"][i, 0] + dff.at[ idx, "shift" +
col_target]
        i = i + 1
else:
    predicciones_prueba_undiff = predicciones_prueba
############################################################
ev_ts_data = create_evaluation_df(predicciones_prueba_undiff,
tensor_datos_prueba, HORIZON, scaler_cantidad,
                                  log)
for index, value in ev_ts_data["prediccion"].iteritems():
    if value < DELTA:
        ev_ts_data.at[index,"prediccion"] = DELTA

producto.mape = mape(ev_ts_data["actual"], ev_ts_data["prediccion"])

resultados.append(
    [
        nombre_modelo,
        EPOCHS,
        ("Log " if log else "noLog") +("Diff" if diff else "noDiff"),
        producto.producto_id,
        producto.nombre,
        producto.mape,
    ]
)
```

3.4.6. Selección del mejor modelo

Para poder seleccionar el mejor modelo, se realizó iteraciones de las siguientes variables:

- Épocas: 16,32,64 y 128
- Escalado de datos a través de la función logaritmo
- Transformación de datos a través de la función diff (diferencia discreta)
- Modelos: SVR, GRU,LSTM y CONV

Los datos se guardarán en un archivo llamado resultado.csv para su posterior análisis. La implementación fue la siguiente:

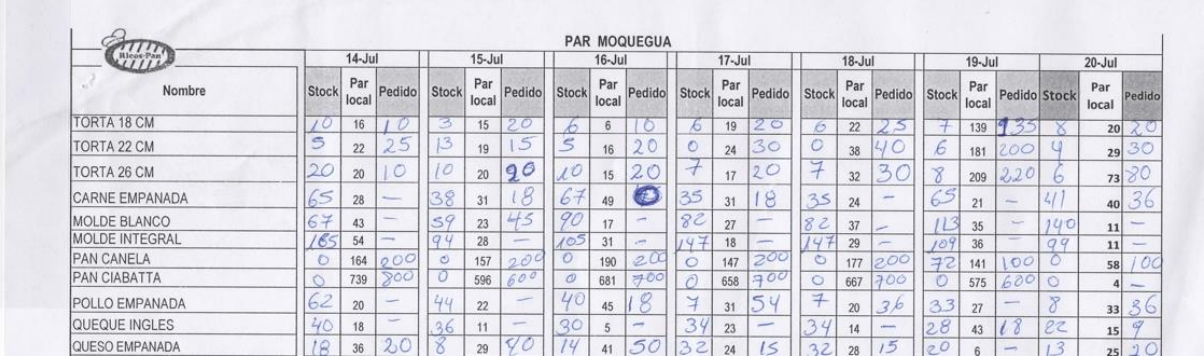
```
col_target = "cantidad"
lstLogDiff = [[True, True], [True, False], [False, False], [False, True]]

resultados = []
resultados.append(["Modelo", "EPOCHS", "LogDiff", "PID", "Nombre", "mape",
"smape", "mae", "mse"])
epocas = [16, 32, 64, 128]
for modelo in modelos:
    for logDiff in lstLogDiff:
        if modelo.nombre == "SVR":
            res = EntrenarProbar(
                modelo,
                producto_datos,
                "2021-07-01",
                columnas_validas=columnas_validas,
                col_target=col_target,
                EPOCHS=0,
                BATCH_SIZE=BATCH_SIZE,
                T=T,
                log=logDiff[0],
                diff=logDiff[1]
            )
            resultados.extend(res)
        else:
            for EPOCHS in epocas:
                res = EntrenarProbar(
                    modelo,
                    producto_datos,
                    "2021-07-01",
                    columnas_validas=columnas_validas,
                    col_target=col_target,
                    EPOCHS=EPOCHS,
                    BATCH_SIZE=BATCH_SIZE,
                    T=T,
                    log=logDiff[0],
                    diff=logDiff[1]
                )
                resultados.extend(res)
```

```
df_resultado = pd.DataFrame(resultados, columns=resultados[0])
df_resultado = df_resultado.drop([0])
df_resultado.to_csv('resultado.csv')
```

3.4.7. Pronostico manual

Los datos del pronóstico manual se extrajeron de los formatos proporcionados por la empresa, estos fueron llenados por el supervisor de tienda como se muestra en la Figura 32, en donde *stock* se refiere al stock final de productos del día anterior, el *par local* se refiere a la venta del anterior mes y el *pedido* a la proyección manual que se hace de acuerdo a los dos datos anteriores.

Figura 32: Hoja de pares para producción



PAR MOQUEGUA

Nombre	14-Jul			15-Jul			16-Jul			17-Jul			18-Jul			19-Jul			20-Jul		
	Stock	Par local	Pedido	Stock	Par local	Pedido	Stock	Par local	Pedido	Stock	Par local	Pedido	Stock	Par local	Pedido	Stock	Par local	Pedido	Stock	Par local	Pedido
TORTA 18 CM	10	16	10	3	15	20	6	6	10	6	19	20	6	22	25	7	139	135	8	20	20
TORTA 22 CM	5	22	25	13	19	15	5	16	20	0	24	30	0	38	40	6	181	200	4	29	30
TORTA 26 CM	20	20	10	10	20	90	10	15	20	7	17	20	7	32	30	8	209	220	6	73	30
CARNE EMPANADA	65	28	—	38	31	18	67	49	0	35	31	18	35	24	—	65	21	—	41	40	36
MOLDE BLANCO	67	43	—	59	23	45	90	17	—	82	27	—	82	37	—	113	35	—	140	11	—
MOLDE INTEGRAL	165	54	—	94	28	—	105	31	—	147	18	—	147	29	—	109	36	—	99	11	—
PAN CANELA	0	164	200	0	157	200	0	190	200	0	147	200	0	177	200	72	141	100	0	58	100
PAN CIABATTA	0	739	800	0	596	600	0	681	700	0	658	700	0	667	700	0	575	600	0	4	—
POLLO EMPANADA	62	20	—	44	22	—	40	45	18	7	31	54	7	20	36	33	27	—	8	33	36
QUEQUE INGLES	40	18	—	36	11	—	30	5	—	34	23	—	34	14	—	28	43	18	22	15	9
QUESO EMPANADA	18	36	20	8	29	80	14	41	50	32	24	15	32	28	15	20	6	—	13	25	20

Fuente: Ricos Pan

Estos datos fueron tabulados como se muestra en la Tabla 7 en donde está compuesto por la fecha, producto, predicción manual y la venta real.

Tabla 7: Predicción manual vs venta real

Fecha	Producto	Predicción manual	Venta Real
1/07/2021	CARNE EMPANADA	51	43
1/07/2021	MOLDE BLANCO	73	41
1/07/2021	MOLDE INTEGRAL	154	50
1/07/2021	PAN CANELA	200	163
1/07/2021	PAN CIABATTA	600	634
1/07/2021	POLLO EMPANADA	30	40
1/07/2021	QUEQUE INGLES	35	17
1/07/2021	QUESO EMPANADA	35	50

Fuente: Adaptado de los registros de Ricos Pan

Después de tabular los datos se procedió a calcular el rendimiento del pronóstico a través de la función del error porcentual absoluto medio absoluta (`mape`), el código para dicho calculo se muestra a continuación:

```
import pandas as pd
from common.utils import  mape
df=pd.read_excel('comparacion_manual.xlsx')
df['Fecha']=pd.to_datetime(df['Fecha']);
df=df[(df['Venta']!=0) | (df['Manual']!=0)] #No se incluyen valores cero
df=df[(df['Codigo']!=3352)&(df['Codigo']!=3354) ] # No se incluyen todos los
productos
print("MAPE:"+str(mape(df['Venta'],df['Manual'])))
```

Cabe destacar que no se incluyeron los datos cuyo valor era cero tanto de las ventas como la predicción manual debido a la restricción de la función de error porcentual absoluto medio. Por otro lado, tampoco se incluyó el molde blanco y el molde integral ya que estos se producen en cantidades mayores, ya que su fecha de vencimiento es de más duración, a comparación de los demás productos.

CAPITULO IV

RESULTADOS Y DISCUSIÓN

Después de preprocesar los datos, implementar el modelo y ejecutar las pruebas correspondientes, se presenta las tablas y figuras para el análisis e interpretación de los datos mostrados para el pronóstico de ventas de los productos de la muestra, cabe señalar que los valores mostrados son la medida de desempeño que muestra cada modelo, esta medida de desempeño está dada por el error porcentual medio absoluto. Los datos para el entrenamiento fueron desde 15/01/2017 al 01/07/2021, y los datos de prueba fueron del 02/07/2021 al 30/11/2021. Para una mejor comprensión de los resultados se muestran en 3 partes, resultados sin preprocesamiento de datos, resultados con procesamiento de datos y la predicción manual que se hace dentro de la empresa Ricos Pan.

4.1. RESULTADOS SIN PREPROCESAMIENTO DE DATOS

Se realizo una prueba sin realizar el preprocesamiento de los datos de los modelos Conv1D, GRU, LSTM y SVR, se probó distintas épocas de entrenamiento (16,32,64,128), no se agregaron características generales, tampoco se hizo un tratamiento de valores atípicos, solo se consideró la cantidad y precio del producto, con el objetivo de medir impacto que tiene el preprocesamiento de datos, se consideró los valores mínimos del error porcentual medio (MAPE) que llego cada modelo con cada producto, el resumen de los resultados se muestran en la Tabla 8.

Tabla 8: Error porcentual absoluto medio de predicción sin preprocesar los datos

Modelo	Epocas	Productos	MAPE
GRU	16	TORTA 18 CM	0.155
LSTM	128	TORTA 22 CM	0.162
Conv1D	64	TORTA 26 CM	0.136
GRU	32	CARNE EMPANADA	0.221
Conv1D	16	MOLDE BLANCO	0.170
GRU	16	MOLDE INTEGRAL	0.179
SVR	0	PAN CANELA	0.242
Conv1D	128	PAN CIABATTA	0.259
GRU	64	POLLO EMPANADA	0.237
GRU	16	QUEQUE INGLES	0.163
Conv1D	128	QUESO EMPANADA	0.277
LSTM	16	TORTA CHOCOLATE ESP. ENTERA	0.085
GRU	16	TORTA HELADA	0.136
GRU	16	TORTA SELVA NEGRA ENTERA	0.115
GRU	16	TORTA TRES LECHES CLASICA ENTERA	0.133
		Promedio	0.1780

Elaboración propia

De acuerdo a la Tabla 8 tenemos un rendimiento de pronóstico en promedio de 0.1780 de acuerdo a MAPE, por otro lado, podemos observar que el modelo GRU es el mejor modelo ya que de los 15 productos 8 fueron tuvieron mejores resultados, seguido del modelo Conv1D con 4 productos, LSTM con 2 y por último 1 con el modelo SVR.

4.2. RESULTADOS CON PREPROCESAMIENTO DE DATOS

Se realizo la prueba con los datos preprocesados, como eliminación de ceros, valores atípicos, y agregando características. Para determinar el mejor modelo se puso a prueba los modelos: Conv1D, LSTM, GRU, y SVR. Se probó distintas épocas de entrenamiento: 16, 32, 64 y 128; por último, también se usó la transformación de datos a través de logaritmo y diferencia.

- noLognoDiff: Se uso una transformación de escala normal min max.
- noLogDiff: Se uso una transformación de diferencia, mas no logarítmica
- LogDiff: Se uso una transformación de logaritmo y diferencia.

- LognoDiff: Se uso una transformación de logaritmo, mas no se usó una transformación de diferencia.

En la Tabla 9 se muestra los resultados de acuerdo a la medida de desempeño de cada modelo, también se muestra todas las épocas de entrenamiento, también las distintas transformaciones que se realizó en los datos de venta del producto TORTA 18 CM.

Tabla 9: Resultados de entrenamiento y prueba del producto Torta 18 cm

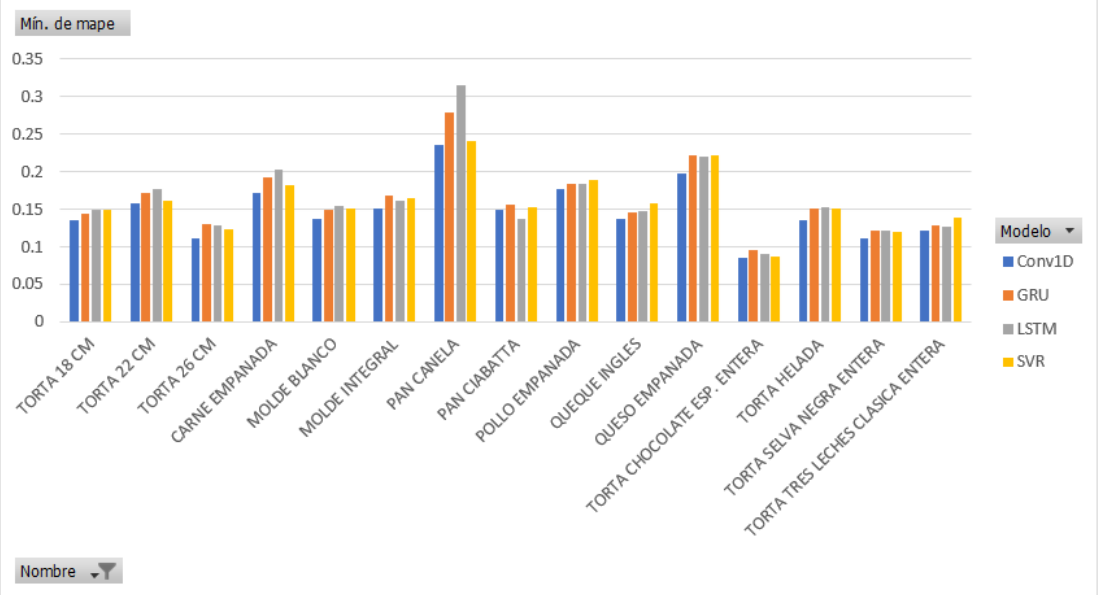
TORTA 18 CM	ÉPOCAS				
Modelo	**0**	**16**	**32**	**64**	**128**
Conv1D					
noLognoDiff		0.339	0.140	0.136	0.140
noLogDiff		0.192	0.151	0.148	0.146
Log Diff		0.159	0.166	0.145	0.157
Log noDiff		0.254	0.180	0.171	0.161
GRU					
Log noDiff		0.163	0.160	0.145	0.168
Log Diff		0.148	0.156	0.192	0.195
noLogDiff		0.303	0.257	0.290	0.296
noLognoDiff		0.578	0.249	0.331	0.364
LSTM					
Log noDiff		0.198	0.173	0.174	0.193
Log Diff		0.149	0.160	0.174	0.207
noLogDiff		0.311	0.311	0.286	0.360
noLognoDiff		0.810	0.258	0.329	0.372
SVR					
Log noDiff	0.150				
noLogDiff	0.155				
Log Diff	0.150				
noLognoDiff	0.160				

Elaboración propia

De acuerdo a la Tabla 9 el valor más óptimo de acuerdo a la medida de desempeño es de 0.1362 que corresponde al modelo Conv1d con 64 épocas de entrenamiento y aplicando una transformación de escala normal min max.

De igual manera se analizó para todos los productos de la muestra obteniendo los resultados como se muestran en la Figura 33, siendo los valores mostrados en las barras el desempeño del modelo, el detalle de los valores se puede observar en la Tabla 10.

Figura 33: Resumen del error porcentual medio absoluto



Elaboración propia

Tabla 10: Resumen del desempeño de cada modelo por producto.

Modelo	EPOCHS	LogDiff	Nombre	MAPE
Conv1D	64	noLognoDiff	18 CM TORTA	0.136
Conv1D	16	noLognoDiff	22 CM TORTA	0.158
Conv1D	64	noLognoDiff	26 CM TORTA	0.112
Conv1D	16	noLognoDiff	CARNE EMPANADA	0.172
Conv1D	128	noLognoDiff	MOLDE BLANCO	0.137
Conv1D	32	Log Diff	MOLDE INTEGRAL	0.151
Conv1D	32	noLogDiff	PAN CANELA	0.237
LSTM	16	Log Diff	PAN CIABATTA	0.137
Conv1D	16	noLognoDiff	POLLO EMPANADA	0.177
Conv1D	64	noLognoDiff	QUEQUE INGLES	0.138
Conv1D	32	noLognoDiff	QUESO EMPANADA	0.198
Conv1D	32	noLognoDiff	TORTA CHOCOLATE ESP. ENTERA	0.085
Conv1D	128	noLognoDiff	TORTA HELADA	0.136
Conv1D	64	noLogDiff	TORTA SELVA NEGRA ENTERA	0.112
Conv1D	128	noLogDiff	TORTA TRES LECHES CLASICA ENTERA	0.121
			Promedio	**0.147**

Elaboración propia

En la Tabla 11 se muestra el mejor modelo por cada producto de acuerdo a la medida de desempeño, de la misma manera se muestra el número de épocas, siendo el mejor modelo el Conv1D que es una red neuronal convolucional con un total de 14 productos, que representa 93.3%, seguido del modelo LSTM con 1 producto que representa el 6.6%, del total de la muestra de productos. El promedio de la medida de

desempeño de los modelos es de 0.147 a comparación de los resultados sin preprocesamiento de datos que el promedio es de 0.178 existe un 21% de mejora.

4.3. RESULTADOS DEL PRONÓSTICO MANUAL

Se extrajo los datos de pronóstico manual de acuerdo al formato de la Figura 32, en donde se comparó con la venta real de acuerdo con la medida de desempeño, los resultados se muestran en la Tabla 11.

Tabla 11: Resultados de la medida de desempeño del pronóstico manual

PRODUCTO	MAPE
TORTA 18 CM	0.422
TORTA 22 CM	0.350
TORTA 26 CM	0.326
CARNE EMPANADA	0.371
PAN CANELA	0.399
PAN CIABATTA	0.248
POLLO EMPANADA	0.425
QUEQUE INGLES	0.540
QUESO EMPANADA	0.525
TORTA CHOCOLATE ESP. ENTERA	0.470
TORTA HELADA	0.453
TORTA SELVA NEGRA ENTERA	0.451
TORTA TRES LECHES CLASICA ENTERA	0.399
Promedio	**0.414**

Elaboración propia

De acuerdo a la Tabla 11 el peor rendimiento fue del queque inglés con 0.5402 y la empanada de queso con 0.525. En total se observa un promedio de 0.414 a comparación con los resultados obtenidos con los modelos basados en Machine Learning tienen un promedio de 0.147 de acuerdo a la Tabla 10, existe una diferencia de 0.266 que significa una mejora del 35%.

4.4. DISCUSIÓN

A partir de los resultados encontramos aceptamos la hipótesis que señala que un modelo basado en Machine Learning optimiza el pronóstico de ventas en la empresa Ricos Pan año 2020 – 2021.

En cuanto al preprocesamiento de datos tal como lo señala (Hernández & Rodríguez, 2008) y (García et al., 2016) sostienen que el procesamiento de datos la mejoran la eficacia de los modelos, esto es acorde con lo que en este estudio encuentra, ya que de acuerdo a los resultados preprocesar los datos mejoraron la precisión del pronóstico de los modelos en un 20%.

En lo que respecta a la selección del modelo el presente estudio encontró que las redes convolucionales tuvieron un mejor desempeño frente a los otros modelos, no se encontraron estudios donde comparen los modelos propuestos que realicen el pronóstico de ventas en el área de panadería, sin embargo se encontraron otros estudios de otras áreas como lo señala (Ensafi et al., 2022) que realizaron el pronóstico de ventas en una tienda minorista de muebles indicando que el modelo LSTM es mejor que el modelo CNN, lo cual difiere con el presente estudio, ello puede ser la diferencia de rubro o también que usaron una red LSTM apilada. Otro estudio de (Badal & Franzén, 2019) en donde tratan de predecir el precio de la electricidad mediante los modelos SVM y RNN encuentran que el modelo SVM tiene mejores resultados en precisión, lo cual difiere del presenta estudio. (Zhao & Wang, 2017). Por otro lado (Zhao & Wang, 2017) hizo un estudio de pronóstico de ventas del sitio web cianao.com utilizando CNN en donde los resultados que obtuvieron concuerdan con las del presente este estudio. (Osowski et al., 2004) encontró que las redes MLP tienen mejor habilidad de generalización en cuanto a regresiones, esto concuerda con los resultados hallados en el presente estudio.

Por otro lado, en cuanto a la comparación del pronóstico manual y el modelo baso en Machine Learning, se encontró que los modelos basados en Machine Learning fueron muy superiores a los pronósticos manuales que usa la empresa ello corresponde a lo que indica (Aronsson & Jonsson, 2008) en donde señala que este tipo de método manual a menudo son opiniones sesgadas y limitadas.

V. CONCLUSIONES

Primera: En esta investigación se desarrolló un modelo basado en Machine Learning que mejora el pronóstico de ventas de la empresa Ricos Pan, para ello se comparó modelos basados en neuronales recurrentes, redes neuronales convolucionales y una máquina de soporte vectorial, concluyendo que la red neuronal convolucional tuvo mejores resultados que los otros métodos, este modelo optimizó la predicción de ventas en un 35% a diferencia de la predicción manual que se hace de forma convencional en la empresa.

Segunda: Se preprocesó los datos utilizando diversas técnicas, este preprocesamiento mejoró la precisión de pronóstico de los modelos en un 21% en promedio, los datos fueron extraídos del Sistema de ventas de la empresa, se trataron los valores atípicos mediante el diagrama cajas bigotes y mediante rangos intercuantiles, de la misma forma se identificaron datos faltantes que se debieron a que la empresa cerró por pandemia Covid – 19, por otro lado también se identificó ventas atípicas por días festivos como el día del padre, día de la madre, fines de semana estos valores se trataron agregando atributos a los datos.

Tercera: Se implementaron los modelos basados en redes neuronales convolucionales (CONV), redes neuronales recurrentes (LSTM y GRU) y un modelo basado en máquina de soporte vectorial (SVR), en el lenguaje Python se utilizó diversos frameworks como Tensor Flow, Keras y scikitlearn; los modelos CONV, LSTM, y GRU aceptan la adición de características a los datos como el de días festivos, días en que la empresa cerró por pandemia Covid – 19 y fines de semana pero el modelo SVR no es compatible con esta característica, por lo que se implementó una clase para que dichos modelos puedan interactuar entre sí con el fin de entrenarse y probarse en conjunto, por

otro lado se implementó una función para entrenar y probar los modelos en donde se establecieron parámetros para su entrenamiento como épocas, transformación de datos mediante logaritmos y diferencia simple, también se adiciono un parámetro de fecha para separar los datos de entrenamiento y prueba.

Cuarta: Se evaluó y se determinó que el modelo más optimo es el modelo denominado CONV que está basado en redes neuronales convolucionales que representa un 93.3% de la muestra, mientras que el modelo LSTM ocupo un segundo lugar con un 6.6%, para llegar a dicha conclusión se entrenaron los modelos y se probaron con distintas épocas de entrenamiento, cada producto se entrenó con un numero de épocas distintas que van desde 16 hasta los 128, se observó que un 66.6% de la muestra no necesitó transformaciones de logaritmo o diferencia simple, mientras que el 33.3% si las necesitó.

Quinta: Se comparó el pronóstico de ventas manual que hace la empresa con el modelo seleccionado y se concluye que el modelo basado en Machine Learning, supera en un 35% en promedio de acuerdo a la medida de desempeño.

VI. RECOMENDACIONES

Se recomienda para trabajos posteriores relacionados a la predicción de ventas en el utilizar redes CNN y LSTM en vista de que ambos modelos mostraron buenos resultados a la hora de hacer pronósticos de venta teniendo en cuenta los datos históricos.

Se recomienda preprocesar los datos antes de entrenar los modelos basados en Machine Learning, a partir de las técnicas de preprocesamiento vistas en la presente tesis, para futuras investigaciones se recomienda hacer el uso de la correlación entre las características de los datos para obtener una mejor precisión del pronóstico de los modelos, es decir para cierto producto se debe incluir ciertas características.

Se recomienda que al momento de implementar la clase para entrenar y probar los modelos se deba agregar un parámetro en donde estén presentes algunas características de acuerdo al preprocesamiento de datos, ya que en la presente tesis se consideraron todas.

VII. REFERENCIAS BIBLIOGRÁFICAS

Ahmed, N. K., Atiya, A. F., el Gayar, N., & El-Shishiny, H. (2010). An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews*, *29*(5), 594–621. https://doi.org/10.1080/07474938.2010.481556

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2018). Understanding of a convolutional neural network. *Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017*, 1–6. https://doi.org/10.1109/ICEngTechnol.2017.8308186

Anthony, R., & Govindarajan, V. (2011). *Control Systems*. McGraw-Hill.

Aronsson, H., & Jonsson, R. (2008). Sales forecasting management. *School of Bussines, University of Gothenburg*.

Augusto, E., Santoyo, R., Antonio, J., & González, L. (2014). Comparación de predicción basada en redes neuronales contra métodos estadísticos en el pronóstico de ventas. *Ingeniería Industrial. Actualidad y Nuevas Tendencias*, *IV*(12), 91–105.

Badal, L., & Franzén, S. (2019). A Comparative Analysis of RNN and SVM. In *Degree Project Computer Engineering*. KTH Royal Institute of Technology in Stockholm.

Betancourt, G. (2005). Las máquinas de soporte vectorial. *Scientia et Technica*, *27*(27), 67–72.

Bontempi, G., ben Taieb, S., & le Borgne, Y. A. (2013). Machine learning strategies for time series forecasting. *Lecture Notes in Business Information Processing*, *138*(06), 62–77. https://doi.org/10.1007/978-3-642-36318-4_3

Botchkarev, A. (2018). Performance Metrics (Error Measures) in Machine Learning Regression, Forecasting and Prognostics: Properties and Typology. *Journal of Chemical Information and Modeling*, *53*(9), 1689–1699. https://doi.org/10.28945/4184

Botchkarev, A. (2019). A new typology design of performance metrics to measure errors in machine learning regression algorithms. *Interdisciplinary Journal of Information, Knowledge, and Management*, *14*(01), 45–76. https://doi.org/10.28945/4184

Cadena Lozano, J. B. (2016). *Gestión del pronóstico estratégico*. Editorial CESA.

Carbonero Pinto, E. (2021). *Análisis de tráfico de Internet mediante el uso del suavizado exponencial de series temporales* [Tesis de pregrado, Universidad Autónoma de Madrid]. http://zaguan.unizar.es/TAZ/EUCS/2014/14180/TAZ-TFG-2014-408.pdf

Carranza, O., & Sabría, F. (2005). *Mejores prácticas logísticas en Latinoamérica*. International Thomson.

Chatfield, C. (2005). *Statistical Analysis and Data Display*. Springer.

Chicco, D., Warrens, M. J., & Jurman, G. (2021). The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation. *PeerJ Computer Science*, *7*(7), 1–24. https://doi.org/10.7717/PEERJ-CS.623

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *Presented in NIPS 2014 Deep Learning and Representation Learning Workshop*, 1–9. http://arxiv.org/abs/1412.3555

Ensafi, Y., Amin, S. H., Zhang, G., & Shah, B. (2022). Time-series forecasting of seasonal items sales using machine learning – A comparative analysis. *International Journal of Information Management Data Insights*, *2*(1), 100058. https://doi.org/10.1016/j.jjimei.2022.100058

Garcia, S. (2022). Forecasting: Principles and Practice. *Forecasting: Principles and Practice*, *237*. https://rpubs.com/Sergio_Garcia/forecasting_r_part_4

García, S., Luengo, J., & Herrera, F. (2015). *Data Preprocessing in Data Mining*. Springer.

García, S., Ramírez, S. G., Luengo, J., & Herrera, F. (2016). Big Data: Preprocesamiento. *Novática*, *237*, 17–23. http://sci2s.ugr.es/sites/default/files/ficherosPublicaciones/2133_Nv237-Digital-sramirez.pdf

Garzon, I. (2022). *Cómo usar redes neuronales (LSTM) en la predicción de averías en las máquinas*. https://blog.gft.com/es/2018/11/06/como-usar-redes-neuronales-lstm-en-la-prediccion-de-averias-en-las-maquinas/

Hernández, C., & Rodríguez, J. (2008). Preprocesamiento De Datos Estructurados. *Revista Vínculo*, *4*(2), 27–48. https://doi.org/10.14483/2322939X.4123

Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, P. (2014). *Metodología de la investigación* (6ta ed.). McGRAW-HILL.

IBM. (2022). *Transformaciones de series*. https://www.ibm.com/docs/es/spss-modeler/SaaS?topic=data-series-transformations

Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., & Muller, P. A. (2019). Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, *33*(4), 917–963. https://doi.org/10.1007/s10618-019-00619-1

Kelleher, J. D., Namee, B. mac, & D'arcy, A. (2015). *Fundamentals of Machine Learning for predictive data analytics*. MIT Press.

Kim, P. (2017). Neural Network and Classification. *MATLAB Deep Learning*, 81–102. https://doi.org/10.1007/978-1-4842-2845-6_4

KIM, W., CHOI, B.-J., HONG, E.-K., & KIM, S.-K. (2002). A Taxonomy of Dirty Data. *Data Mining and Knowledge Discovery*, *103*(3), 20. https://doi.org/10.1023/A:1021564703268

Kubat, M. (2017). An Introduction to Machine Learning. In *An Introduction to Machine Learning*. Springer. https://doi.org/10.1007/978-3-319-63913-0

Lazzeri, F. (2020). *Machine Learning for Time Series Forecasting with Python®*. Wiley. https://doi.org/10.1002/9781119682394

Luo, Z., Liu, H., & Wu, X. (2005). Artificial neural network computation on graphic process unit. *Proceedings of the International Joint Conference on Neural Networks*, *1*, 622–626. https://doi.org/10.1109/ijcnn.2005.1555903

Maravall, A. (1996). *Unobserved components in economic time series*. European University Library.

Mentzer, J., & Moon, M. (2005). *Sales Forecasting Management: A Demand Management Approach* (2nd ed.). SAGE Publications, Inc. https://doi.org/10.4135/9781452204444

Naghi Namakforoosh, M. (2005). *Metodología de la Investigación* (2da ed.). Limusa.

Nojek, S., Britos, P., Rossi, B., & Martínez, R. G. (2003). PRONÓSTICO DE VENTAS: COMPARACIÓN DE PREDICCIÓN ENTRE REDES NEURONALES Y MÉTODO ESTADÍSTICO. *Revista Eletrônica de Ciência Administrativa*, *2*(1), 1–18. https://doi.org/10.5329/RECADM.20030201008

O'Shea, K., & Nash, R. (2015). An Introduction to Convolutional Neural Networks. *ArXiv*. https://doi.org/10.48550/ARXIV.1511.08458

Osowski, S., Siwek, K., & Markiewicz, T. (2004). MLP and SVM networks - A comparative study. *Report - Helsinki University of Technology, Signal Processing Laboratory*, *46*(2), 37–40. https://doi.org/10.1109/NORSIG.2004.250120

Palit, Ajoy K., Popovic, D. (2005). *Computational Intelligence in Time Series Forecasting*. Springer-Verlag. https://doi.org/10.1007/1-84628-184-9

Parrella, F. (2007). *Online support vector regression* [Tesis posgrado, University of Genoa]. http://wiki.icub.org/images/8/82/OnlineSVR_Thesis.pdf

Sanders, F. (2019). *Python Machine Learning For Beginners: Handbook For Machine Learning, Deep Learning And Neural Networks Using Python, Scikit-Learn And TensorFlow*. Independently published. https://books.google.com.pe/books?id=ckEAEAAAQBAJ

Shewalkar, A., nyavanandi, D., & Ludwig, S. A. (2019). Performance Evaluation of Deep neural networks Applied to Speech Recognition: Rnn, LSTM and GRU. *Journal of Artificial Intelligence and Soft Computing Research*, *9*(4), 235–245. https://doi.org/10.2478/jaiscr-2019-0006

Tatsat, H., Puri, S., Lookabaugh, B., & Safari, an O. M. Company. (2020). *Machine Learning and Data Science Blueprints for Finance*. O'reilly.

Tensorflow. (2022). *Time series forecasting*. https://www.tensorflow.org/tutorials/structured_data/time_series

Velásquez, J. D., Olaya, Y., & Franco, C. J. (2010). Predicción de series temporales usando máquinas de vectores de soporte. *Ingeniare, Revista Chilena de Ingeniería*, *18*(1), 64–75.

Ye, J. (2015). *Machine Learning for Exploratory Data Analysis and Predictive Modeling* [Master's thesis]. 71. http://hdl.handle.net/11250/299605

Zhao, K., & Wang, C. (2017). *Sales Forecast in E-commerce using Convolutional Neural Network*. http://arxiv.org/abs/1708.07946

Zheng, A., & Casari, A. (2018). *Feature Engineering for Machine Learning and Data Analytics*. O'reilly. https://doi.org/10.1201/9781315181080

ANEXOS

Anexo 1. Código fuente

Nombre de archivo: entrenar_probar.py

Descripción: En este archivo se definen los modelos CNN, LSTM, GRU y SVR, también se definen las épocas de entrenamiento, así mismo se define la función `EntrenarProbar` que se usa para entrenar y probar cada uno de los modelos, dentro de sus parámetros están la lista de productos, la fecha que indica el inicio de los datos de prueba, las columnas que indican características y por último si los datos serán transformados a través de logaritmo o diferencia, los resultados se guardan en un archivo llamado **resultados.csv.**

```
import datetime as dt
import pandas as pd
from common.entrenamiento import Modelo
import tensorflow as tf
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.layers import GRU, Dense
from tensorflow.keras.models import Sequential
from sklearn.svm import SVR
import numpy as np
from data import cargarTransformarDatos
from tabulate import tabulate
import matplotlib.pyplot as plt
from tensorflow.python.keras.callbacks import ModelCheckpoint
from common.utils import smape, mape, TimeSeriesTensor, create_evaluation_df
from sklearn.metrics import mean_squared_error, mean_absolute_error
pd.options.display.float_format = "{:,.4f}".format
def EntrenarProbar(modelo,
                   lista_productos,
                   fecha_prueba,
                   columnas_validas,
                   col_target,
                   T,
                   EPOCHS,
                   BATCH_SIZE,
                   log=False,
                   diff=False):
    resultados = []
    nombre_modelo = modelo.nombre
    HORIZON = 1
    DELTA = 10
    for producto in lista_productos:
        mejor_modelo_nombre = \
            'modelos/best_model.' + nombre_modelo + '.' + \
            ("Log" if log else "noLog") + \
            ("Diff" if diff else "noDiff") + \
            '.' + \
            str(producto.producto_id) +\
            '.' + \
            str(EPOCHS) + '.h5'
        dff = pd.DataFrame.copy(producto.datos)
        # transformacion de coordenadas
        dff['cantidad'] = dff['cantidad'] + DELTA
        scaler_cantidad = MinMaxScaler()
        # copia de cantidad
        dff['realCantidad'] = dff['cantidad']
        # escalar datos en caso de que no se seleccione logaritmo
        scaler_cantidad.fit(dff[[col_target]])
        dff['logCantidad'] = np.log(dff['cantidad'])
```

```
        if log:
            dff[col_target] = np.log(dff[col_target])
        else:
            dff[col_target] = scaler_cantidad.transform(dff[[col_target]])

        if diff:
            # necesario para hacer unshift
            dff['shift' + col_target] = dff[col_target].shift(1)
            dff[col_target] = dff[col_target].diff()
            dff = dff.dropna()

        # scaler para otras caracteristicas que no sean el target
        scaler_caracteristicas = MinMaxScaler()
        if(len(columnas_validas)>1) :
            dff[columnas_validas[1:]] =
scaler_caracteristicas.fit_transform(dff[columnas_validas[1:]])

        # DATOS PARA ENTRENAMIENTO
        datos_entrenamiento = dff.copy()[dff.index < fecha_prueba][columnas_validas]
        tensor_structure = {"X": (range(-T + 1, 1), columnas_validas)}
        tensor_datos_entrenamiento = TimeSeriesTensor(
            dataset=datos_entrenamiento,
            target=col_target,
            H=HORIZON,
            tensor_structure=tensor_structure,
            freq="D",
            drop_incomplete=True
        )

        # DATOS PRUEBA
        # fecha pasada= fecha_prueba - dias anteriores


        fecha_pasada = dt.datetime.strptime(fecha_prueba, "%Y-%m-%d") - dt.timedelta(days=T - 1)
        datos_prueba = dff.copy()[(dff.index >= fecha_pasada)][columnas_validas]
        tensor_datos_prueba = TimeSeriesTensor(
            dataset=datos_prueba,
            target=col_target,
            H=HORIZON,
            tensor_structure=tensor_structure,
            freq="D",
            drop_incomplete=True,
        )
        check_point = ModelCheckpoint(mejor_modelo_nombre, monitor='val_mse',
save_best_only=True)
        modelo.entrenar(
            tensor_datos_entrenamiento["X"],
            tensor_datos_entrenamiento["target"],
            batch_size=BATCH_SIZE,
            epochs=EPOCHS,
            validation_data=(tensor_datos_prueba["X"], tensor_datos_prueba["target"]),
            callbacks=[check_point],
            verbose=1,
        )
        predicciones_prueba=modelo.predecir(tensor_datos_prueba["X"],mejor_modelo_nombre)
        # Funcion inversa de diff
        if diff:
            predicciones_prueba_undiff = np.zeros(predicciones_prueba.shape)
            i = 0
            for idx in tensor_datos_prueba.dataframe.index:
                idx = idx + dt.timedelta(days=1)  # es un dia mas
                predicciones_prueba_undiff[i, 0] =\
                    predicciones_prueba[i][0] + dff.at[idx, "shift" + col_target]
                tensor_datos_prueba["target"][i, 0] =\
                    tensor_datos_prueba["target"][i, 0] +\
                    dff.at[idx, "shift" + col_target]
                i = i + 1
        else:
            predicciones_prueba_undiff = predicciones_prueba
        ############################################################
        ev_ts_data = create_evaluation_df(
            predicciones_prueba_undiff,
            tensor_datos_prueba,
            HORIZON,
```

```
            scaler_cantidad,
            log
        )
        for index, value in ev_ts_data["prediccion"].iteritems():
            if value < DELTA:
                ev_ts_data.at[index,"prediccion"] = DELTA

        producto.mape = mape(ev_ts_data["actual"], ev_ts_data["prediccion"])
        producto.smape = smape(ev_ts_data["actual"], ev_ts_data["prediccion"])
        producto.mae = mean_absolute_error(ev_ts_data["actual"], ev_ts_data["prediccion"])
        producto.mse = mean_squared_error(
            ev_ts_data["actual"],
            ev_ts_data["prediccion"],
            squared=False
        )

        resultados.append(
            [
                nombre_modelo,
                EPOCHS,
                ("Log " if log else "noLog") +("Diff" if diff else "noDiff"),
                producto.producto_id,
                producto.nombre,
                producto.mape,
                producto.smape,
                producto.mae,
                producto.mse
            ]
        )
        ev_ts_data[["actual", "prediccion"]] = ev_ts_data[["actual", "prediccion"]] - DELTA

        predicciones_anteriores=modelo.predecir(
            tensor_datos_entrenamiento["X"],
            mejor_modelo_nombre)
        # Undiff datos para entrenamiento predichos
        if diff == True:
            predicciones_anteriores_undiff = np.zeros(predicciones_anteriores.shape)
            i = 0
            for idx in tensor_datos_entrenamiento.dataframe.index:
                idx = idx + dt.timedelta(days=1)
                predicciones_anteriores_undiff[i, 0] = \
                    predicciones_anteriores[i][0] + dff.at[idx, "shift" + col_target]
                tensor_datos_entrenamiento["target"][i, 0] = \
                    tensor_datos_entrenamiento["target"][i, 0] + dff.at[
                    idx, "shift" + col_target]
                i = i + 1
        else:
            predicciones_anteriores_undiff = predicciones_anteriores
        ##############################################################
        predicciones = create_evaluation_df(
            predicciones_anteriores_undiff,
            tensor_datos_entrenamiento,
            HORIZON,
            scaler_cantidad,
            log)
        predicciones[["actual", "prediccion"]] =\
            predicciones[["actual", "prediccion"]] - DELTA

        producto.predicciones_prueba = ev_ts_data
        producto.predicciones_entrenamiento = predicciones
    return resultados


columnas_validas = \
    ["cantidad", "dia_semana", "semana",
     "mes", "year", "dia", "local_cerrado",
     "dias_festivos","precio"]


producto_datos = cargarTransformarDatos()
LATENT_DIM = 64
T = 6
BATCH_SIZE = 32
HORIZON = 1
```

```
modelos = []

model1 = Sequential()

model1.add(GRU(LATENT_DIM, input_shape=(T, len(columnas_validas))))
model1.add(Dense(HORIZON))
model1.compile(optimizer="RMSprop", loss="mse", metrics=['mse'])
modelos.append(Modelo(model1, "GRU"))

model2 = tf.keras.models.Sequential([
    tf.keras.layers.LSTM(LATENT_DIM, return_sequences=False),
    tf.keras.layers.Dense(units=1)
])
model2.compile(optimizer="RMSprop", loss="mse", metrics=['mse'])
modelos.append(Modelo(model2, "LSTM"))

model3 = tf.keras.Sequential([
    tf.keras.layers.Conv1D(filters=32,
                           kernel_size=(T,),
                           activation='relu'
                           ),
    tf.keras.layers.Dense(units=32, activation='relu'),
    tf.keras.layers.Dense(units=1),
])
model3.compile(optimizer="RMSprop", loss="mse", metrics=['mse'])
modelos.append(Modelo(model3, "Conv1D"))

model4 = SVR()
modelos.append(Modelo(model4, "SVR"))


col_target = "cantidad"
#Transformaciones
lstLogDiff = [[True, True], [True, False], [False, False], [False, True]]
#lstLogDiff = [[False, False]]

resultados = []
resultados.append(
    ["Modelo", "EPOCHS", "LogDiff",
     "PID", "Nombre", "mape", "smape",
     "mae", "mse"])
epocas = [16, 32, 64, 128]

for modelo in modelos:
    for logDiff in lstLogDiff:
        if modelo.nombre == "SVR":
            res = EntrenarProbar(
                modelo,
                producto_datos,
                "2021-07-01",
                columnas_validas=columnas_validas,
                col_target=col_target,
                EPOCHS=0,
                BATCH_SIZE=BATCH_SIZE,
                T=T,
                log=logDiff[0],
                diff=logDiff[1]
            )
            resultados.extend(res)
        else:
            for EPOCHS in epocas:
                res = EntrenarProbar(
                    modelo,
                    producto_datos,
                    "2021-07-01",
                    columnas_validas=columnas_validas,
                    col_target=col_target,
                    EPOCHS=EPOCHS,
                    BATCH_SIZE=BATCH_SIZE,
                    T=T,
                    log=logDiff[0],
                    diff=logDiff[1]
                )
                resultados.extend(res)
```

```
df_resultado = pd.DataFrame(resultados, columns=resultados[0])
df_resultado = df_resultado.drop([0])
df_resultado.to_csv('resultado.csv')
```

Nombre de archivo: entrenamiento.py

Descripción: En este archivo se definen la clase `Modelo` en donde se define las funciones entrenar en donde sus parámetros los datos de entrar y salida, el tamaño de batch las épocas, los datos de prueba; por otro lado, se define la función `predecir` que predice los datos que se ingresar a través del parámetro data y retorna los valores predichos, por ultimo las funciones `datosSVRY` y `datosSVRX` sirven para dar compatibilidad entre los modelos, ya que no todos soportan los mismos datos y parámetros de entrada.

```
import datetime as dt
import numpy as np
import tensorflow as tf
class Modelo:
    def __init__(self, model, nombre):
        self.model = model
        self.nombre = nombre
    def entrenar(self,
                 dataX,
                 dataY,
                 batch_size,
                 epochs,
                 validation_data,
                 callbacks,
                 verbose):
        if self.nombre == "SVR":
            Xtrain=self.datosSVRX(dataX)
            Ytrain=self.datosSVRY(dataY)
            self.model.fit(Xtrain,Ytrain)
        else:
            self.model.fit(
                dataX,
                dataY,
                batch_size=batch_size,
                epochs=epochs,
                validation_data=validation_data,
                callbacks=callbacks,
                verbose=verbose,
            )
    def predecir(self,data,mejor_modelo_nombre=""):
        if self.nombre == "SVR":
            return self.model.predict(self.datosSVRX(data)).reshape(-1, 1)
        else:
            mejor_modelo = tf.keras.models.load_model(mejor_modelo_nombre)
            return mejor_modelo.predict(data).reshape(-1, 1)

    def datosSVRX(self, dataX):
        X = []
        T=dataX.shape[1]
        for data in dataX:
            first=data[:,0:1].reshape(1,-1)
            X.append(first)
        return np.array(X).reshape(-1,T)

    def datosSVRY(self,dataY):
        Y = []
        for data in dataY:
            Y.append(data)
        return np.array(Y).ravel()
```

Anexo 2. Formato de pronóstico manual de la Empresa Ricos Pan.

PAR MOQUEGUA

Nombre	14-Jul			15-Jul			16-Jul			17-Jul			18-Jul			19-Jul			20-Jul		
	Stock	Par local	Pedido	Stock	Par local	Pedido	Stock	Par local	Pedido	Stock	Par local	Pedido	Stock	Par local	Pedido	Stock	Par local	Pedido	Stock	Par local	Pedido
TORTA 18 CM	10	16	10	3	15	20	6	6	10	6	19	20	6	22	25	7	139	135	8	20	20
TORTA 22 CM	5	22	25	13	19	15	5	16	20	0	24	30	0	38	40	6	181	200	4	29	30
TORTA 26 CM	20	20	10	10	20	20	10	15	20	7	17	20	7	32	30	8	209	220	6	73	80
CARNE EMPANADA	65	28	—	38	31	18	67	49	60	35	31	18	35	24	—	65	21	—	41	40	36
MOLDE BLANCO	67	43	—	59	23	45	90	17	—	82	27	—	82	37	—	113	35	—	140	11	—
MOLDE INTEGRAL	185	54	—	94	28	—	105	31	—	147	18	—	147	29	—	109	36	—	99	11	—
PAN CANELA	0	164	200	0	157	200	0	190	200	0	147	200	0	177	200	72	141	100	0	58	100
PAN CIABATTA	0	739	800	0	596	600	0	681	700	0	658	700	0	667	700	0	575	600	0	4	—
POLLO EMPANADA	62	20	—	44	22	—	40	45	18	7	31	54	7	20	36	33	27	—	8	33	36
QUEQUE INGLES	40	18	—	36	11	—	30	5	—	34	23	—	34	14	—	28	43	18	22	15	9
QUESO EMPANADA	18	36	20	8	29	40	14	41	50	32	24	15	32	28	15	20	6	—	13	25	20
TORTA CHOCOLATE ESP. ENTERA	1	2	2	1		2	1	1	2	2	2	—	2	1	—	0	8	10	2	1	2
TORTA HELADA	6	9	4	5	8	4	3	3	6	3	3	—	9	14	8	5	13	20	4		—
TORTA SELVA NEGRA ENTERA	2	5	4	3	5	4	2	4	4	0	4	6	0	8	10	0	32	40	6	8	2
TORTA TRES LECHES CLASICA ENTERA	6	11	8	8	6	6	6	9	6	2	9	14	2	8	12	7	51	60	5	2	5