UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

Ingeniería de datos para pronóstico de ventas e integración web/DB de los datos completos de productos en venta

Edison Jair Méndez Quispe

Ingeniería en Ciencias de la Computación

Trabajo de fin de carrera presentado como requisito
para la obtención del título de
Ingeniero en ciencias de la computación

Quito, 19 de mayo de 2023

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

HOJA DE CALIFICACIÓN
DE TRABAJO DE FIN DE CARRERA

Ingeniería de datos para pronóstico de ventas e integración web/DB de los datos completos de productos en venta

Edison Jair Méndez Quispe

Nombre del profesor, Título académico Daniel Fellig Goldvechmiedt, MSc.

Quito, 19 de mayo de 2023

© DERECHOS DE AUTOR

Nombres y apellidos: Edison Jair Méndez Quispe

Código: 214464

Cédula de identidad: 1751251818

Lugar y fecha: Quito, 19 de mayo de 2023

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en http://bit.ly/COPETheses.

UNPUBLISHED DOCUMENT

Note: The following capstone project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on http://bit.ly/COPETheses.

RESUMEN

El presente proyecto integrador tiene como objetivo la maximización de los datos en términos de pronósticos de ventas, para que estos datos sean útiles, ya sea para poder visualizar la información desde un repositorio web, o que sean usados mediante un modelo de aprendizaje supervisado. La idea del proyecto surge de una necesidad de negocios real, y este trabajo del autor es una respuesta a esa necesidad. Esto se debe a que, conforme avanza la tecnología, muchas empresas requieren el uso de metodologías que permitan explotar estos avances al máximo. En este proyecto integrador el autor resolverá problemas de procesamiento, captación de datos, y predicción. El origen de estos datos vendrá de un sistema real y productivo, conocido como MBA3, del cual se extraerá información vía red y vía plantilla.

Palabras clave: SKU, ventas, procesamiento, limpieza de datos, ERP, modelo de aprendizaje profundo (Deep Learning), framework, base de datos, algoritmo, dataset, predicción, redes neuronales, LSTM.

ABSTRACT

The objective of this capstone project is to understand the importance of a company's data, for visualization and supervisory model-training purposes for sales forecasting. The idea of the project arises from a real business need as understood by the author. This is because, as technology advances today, many companies to be competitive require the use of methodologies that exploit these advances to the fullest. In this project, the author will discuss and implement the processing, data capture, and prediction/forecasting problems. The origin of these data will come from a real and productive system, known as MBA3, from which the information will be extracted through the network.

Keywords: SKU, sales, processing, data cleansing, ERP, Deep learning, framework, database, algorithm, dataset, forecasting, neuronal networks, LSTM.

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Conforme avanza la tecnología en la sociedad, las personas y/o empresas deben estar aptas para adaptarse a lo que su entorno lo demanda, buscar la mejora continua y ser capaces de dar una solución ante cualquier problema que se presente. Para el presente proyecto se manejan dos problemáticas: la falta de información que un vendedor tiene para vender su producto; y el manejo, procesamiento, limpieza y recolección de datos, para que estos sirvan y se apliquen modelos de predicción; en este caso se plantea predecir las ventas de la compañía en base a sus datos históricos. Dichos problemas se abordarán desde dos puntos: desarrollo web con doble integración a bases de datos relacionadas a toda la información del producto; y mediante el uso de técnicas de aprendizaje supervisado.

El origen de las problemáticas descritas anteriormente, se presentan como una necesidad real. En el campo laboral que actualmente el autor se encuentra, los jefes de área han planteado la necesidad de resolver dichos problemas, con la esperanza de tener mayor flexibilidad y manejo de información para poder realizar sus ventas. Así como también poseer datos de calidad que permitan un mejor control de la información correspondiente a sus ventas históricas.

A lo largo del desarrollo del proyecto integrador, se usará con frecuencia palabras como: ERP y MBA3, por lo que es importante mencionar definir cada uno de estos términos. ¿Qué es un ERP? Un ERP no es más que un software especializado en ventas y producción; para el proyecto integrador, se considerará únicamente un ERP orientado a ventas. ¿Qué es MBA3? Es un ERP que tiene conexión cliente – servidor y que ha sido de uso por varios años en la empresa en donde actualmente se trabaja; su principal uso es por parte de los vendedores con el módulo de punto de venta. Sin embargo, con el uso de este ERP, dichos vendedores han sido expuestos a falta de información que día a día se tiene que actualizar y que

lamentablemente no se ha podido integrar en dicho sistema. Tal y como se explicó anteriormente, esa es una de las problemáticas que se pretende resolver mediante el uso de desarrollo web, usando el framework de Django.

¿Qué es un SKU? Un código de referencia, o "stock-keeping unit" por sus siglas, es un número de referencia **único** que sirve para identificar un elemento o artículo de un inventario físico. Desde este momento en adelante, será necesario referirse a un producto con estas siglas.

Objetivos

a. Generales

Mostrar información más detallada al vendedor de la información de un SKU; información que se pretende sea actualizada con la base de datos del ERP que utiliza.

Captar, procesar, limpiar y compilar varios documentos de ventas históricas de la empresa sobre la que se hace la investigación, para que estos mismos datos sean usados para poder predecir las ventas de la compañía.

b. Específicos

Reemplazar el uso tradicional de tablas de Excel y reemplazar el trabajo humano por un sistema automatizado.

Usar los datos históricos previamente procesados, en el intervalo de tiempo comprendido entre 2017 al 2022, para así poder predecir las ventas que la empresa podría tener el año siguiente. Esto se logrará amaestrando redes neuronales LSTM con datos de la empresa.

DESARROLLO DEL TEMA

1. Estudio del Arte

1.1. Qué es un SKU

Según del Tena, un SKU es un: "*código que identifica a un producto concreto que está a la venta. Un código que, de hecho, está formado por un conjunto de caracteres que es único y que se identifica de forma exclusiva con el producto al que representa*" (2020). Este término será sumamente importante en el desarrollo del proyecto integrador, ya que de esta forma se podrá identificar, buscar y realizar todos los análisis sobre un determinado código. En la primera parte del proyecto integrador se hace precisamente esto, una búsqueda profunda por código o nombre del producto, para que así cualquier persona tener la información a la mano y así vender el producto con mayor facilidad. En el desarrollo se harán búsquedas de dichos códigos o productos hacia una base de datos, la misma que está publicada en una IP pública. Dicha base de datos es la base de datos de producción de la empresa en la que actualmente se trabaja, por lo que las consultas que se hagan a la base de datos reflejan datos actualizados en tiempo real.

El término SKU para empresas como la descrita en este documento es de mucha utilidad, porque se puedo usar para:

- Optimizar la gestión del catálogo del producto
- Tener un mejor seguimiento de inventarios
- **Analizar tendencias de ventas y comportamientos en los vendedores**

Estos códigos SKU son dependientes de cada empresa, y en este caso identificar a los productos bajo ese nombre será de gran utilidad, sobre todo, en el primer enfoque que tiene el

proyecto integrador: búsqueda de información de un código SKU en la web. De esta forma será fácil para un vendedor identificar un producto cuando lo busque desde este repositorio web.

1.2. Redes neuronales

En el mundo de la inteligencia artificial existen muchísimas familias de algoritmos que, dependiendo del problema que se pretenda resolver, unas resultarán más eficaces que otras. Una de ellas es: las redes neuronales. De hecho:

> Las redes neuronales son más que otra forma de emular ciertas características propias de los humanos, como la capacidad de memorizar y de asociar hechos. Si se examinan con atención aquellos problemas que no pueden expresarse a través de un algoritmo, se observará que todos ellos tienen una característica en común: la experiencia (Matiach, 2001, p.1).

Como se describió anteriormente, las redes neuronales en esencia tratan de emular el poderoso mecanismo de aprendizaje que tienen las neuronas de los seres humanos, en algoritmos desarrollados en programación. Cada familia de redes se especializa en un tipo de tarea. Por ejemplo, las redes neuronales LSTM (long short-term memory) son ampliamente conocidas y utilizadas para: predicción, clasificación de características, etc. Para el caso que se presenta en este proyecto integrador, se usarán este tipo de redes neuronales ya que resultan ser sumamente eficientes en problemas que tengan que ver con series de tiempo con objetivo de predicción. Más adelante se dará más detalles de este tipo de redes.

1.2.1. Redes neuronales LSTM

Las redes neuronales LSTM son redes recurrentes usadas en el contexto de Deep Learning (aprendizaje profundo), que son capaces de mostrar resultados bastante precisos en problemas de predicción de datos. La problemática que se plantea en la segunda parte del

proyecto integrador es, predecir los volúmenes de ventas de la compañía en el año 2023 usando como datos históricos, las ventas obtenidas en el período comprendido entre 2017 y 2022.

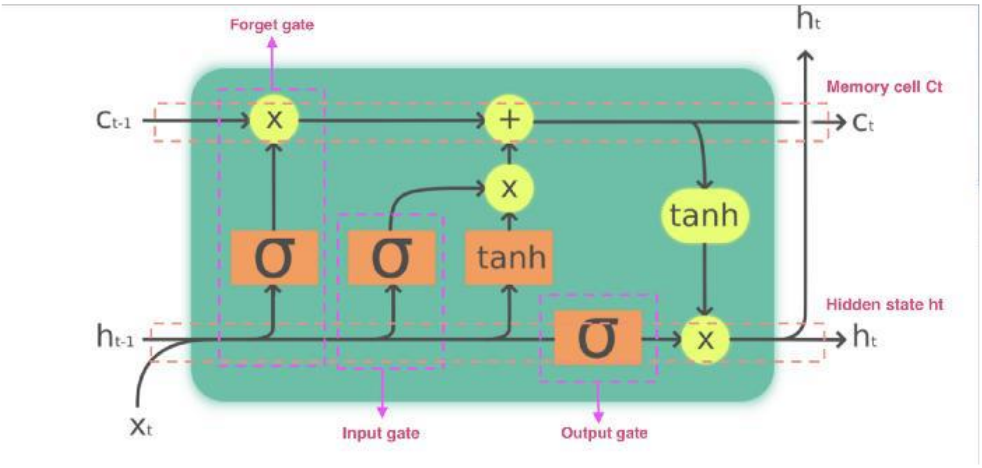
¿Qué son y para qué sirven las redes neuronales LSTM? Según Muñoz y Salazar, las redes LSTM son:

> Un tipo especial de redes recurrentes cuya característica principal es que la información puede persistir introduciendo bucles en el diagrama de la red, por lo que, básicamente, pueden "recordar" estados previos y utilizar esta información para decidir cuál será el siguiente. Esta característica las hace muy adecuadas para manejar series cronológicas (Sandoval y Vallejos, 2022).

Este tipo de red neuronal es la más adecuada para resolver la problemática de predicción descrita anteriormente, a la que se le va a amaestrar con los datos históricos mencionados. Estos datos se usarán como "test" y "train". Estos datos comprenden el tiempo (de 2017 a 2022). Esta red neuronal tiene 4 componentes principales:

- **Input gate:** controla qué información debe ingresar, para no ingresar información con ruido, ej.: stop words.
- **Memory cell:** controla que el valor se deba actualizar o eliminar. Almacena información que puede reutilizarse después.
- **Output gate:** controla la información de aprendizaje útil del estado actual de la celda como salida.
- **Forget gate:** controla la eliminación de la información que ya no es necesaria para que siga aprendiendo la red LSTM. Optimiza el rendimiento de la red neuronal.

Ilustración 1: funcionamiento de una red neuronal LSTM



Lo interesante de este tipo de red neuronal es que son especialmente útiles cuando se trabaja con secuencias de datos, ya que pueden aprender y recordar dependencias a largo plazo en los datos secuenciales. Son capaces de mantener y actualizar estados internos a lo largo de la secuencia, lo que les faculta capturar y modelar patrones temporales complejos.

2. Desarrollo: Mostrar información requerida en proyecto web

2.1. Integración con la base de datos SQL Server

Para resolver la primera parte del proyecto integrador se realizó una aplicación web que tiene como objetivo mostrar información en detalle de un producto o SKU (stock keeping unit) para así facilitar al vendedor a tener más información de la que el ERP no le permite tener. En este desarrollo se utilizó como recursos:

- Uso de Framework Django (con Python)
- Conexión a base de datos MBA3 (vía ODBC)
- Uso de base de datos EXCEL
- Uso de PostgreSQL 14 como orígenes de datos
- Computador personal

2.2. Conexión vía ODBC a la base de datos de MBA3

Para extraer la información de la base de datos del sistema MBA3 fue necesario primero instalar un software para origen de datos 4D v19 ODBC Driver 64-bits. Con ello, pasando los parámetros de conexión de autenticación de ODBC, se estableció la conexión mediante una IP Pública.

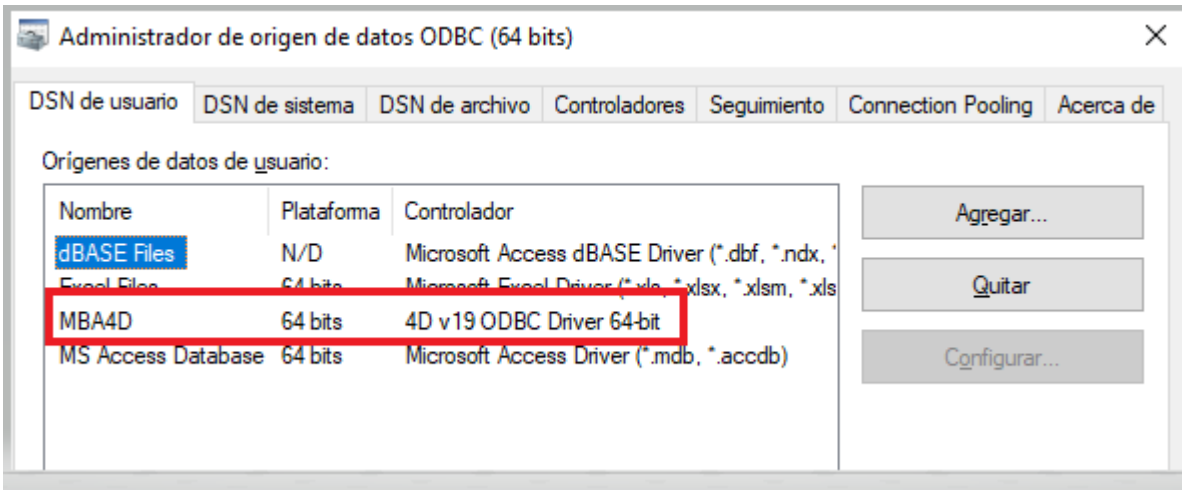


Ilustración 2: Realizando la conexión a base de datos SQL Server

2.3. Realizando la conexión de la base de datos en Python – Django

Una vez realizado los parámetros de configuración inicial para una conexión ODBC en Windows, el sistema operativo ya estuvo listo para establecer la conexión desde Python. Al ser una conexión de base de datos **real** se optó por esconder los parámetros de conexión, para así proteger la integridad de los datos. De esta forma se pudo establecer una correcta conexión a la base datos, la cual ya estuvo lista para empezar a recibir consultas. La consulta principal hacia la base de datos fue únicamente en relación con estos campos:

- Nombre del producto
- Código del producto
- Cantidad disponible del producto

De esta forma se capturó los valores de la base de datos actualizados en tiempo real. Luego, se usó una pequeña base de datos realizada en Excel proporcionada por el actual equipo de trabajo, que contiene la información de los códigos SKU **con sus respectivos números de parte.**

PRODUCTO	CODIGO	N_PARTE
EMBRAGUE DE 1/2 HP GE2387	3	REP8995
TAPA VIDRIO 22 OP 3	4	REST4420
TARJETA CONTROL 16	18	SNS5780
TARJETA CONTROL 16	18	DIR4745
CLIP PARA VENTILADOR 444	48	REP11396
LIBRE EMPAQUE WPL 49CMX1	56	PARTLAV2237
MOTOR LAVADORA000	100	DIR4174
MOTOR EVAPORADOR ORIGINAL DE	101	PART1044
MOTOR ORIGINAL REFRIGERADO	101	ETT4360

Ilustración 3: Información en BD Excel

Y aquí viene la importancia de todo esto, ya que actualmente en el ERP, MBA3, esta información no se muestra, lo que obliga a que vendedor busque el código en Excel para así poder identificar el producto correctamente. A raíz de este problema, surge la idea de crear un repositorio web que junte toda esta información en un solo lugar, para así evitar que el vendedor busque esa información de forma manual e ineficiente. Para realizar una

correcta indexación de la información obtenida en la base de datos SQL Server de MBA3, y la información proporcionada en la pequeña base de datos en Excel, se pasó todos los elementos por separado y se realizó una comparación mediante el uso de diccionarios. Para ello, se hizo comparaciones en las que se usó como clave o *key* el código del SKU, y como valor: nombre del producto, cantidad disponible, y su número de parte.

```
diccionario1 = dict(zip(cod, prodIdBD))
diccionario2 = dict(zip(cod, prodNameBD))
diccionario3 = dict(zip(cod, prodAvailableBD))
diccionario4 = dict(zip(cod, nParteArr))
```

Ilustración 4: Creación de diccionarios

2.4 Almacenando la información recopilada en nueva base de datos

Como se explicó anteriormente, se utilizó diccionarios para indexar la información obtenida tanto de la base de datos de MBA3, como de Excel. Una vez que esa información fue comparada y correctamente relacionada, se procedió a registrar cada uno de los SKU's con su respectiva información en una nueva base de datos, en este caso se usó PostgreSQL. Para ello, en el proyecto Django, en la configuración ***settings.py*** se definió el origen de datos.

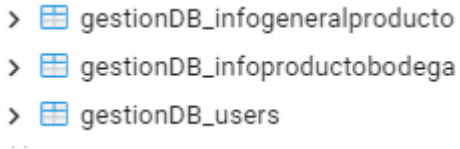
Una vez realizado lo mencionado, en el archivo ***models.py*** ubicada en la aplicación ***ConexionDB*** dentro del proyecto; se procedió a crear las siguientes tablas en la base de datos, las mismas que fueron creadas en PostgreSQL.

Ilustración 5: creación de columnas en la base de datos PostGreSQL

```python
from django.db import models

class Users(models.Model):
    nombre= models.CharField(max_length=30)
    passwords = models.CharField(max_length=30)

class infoGeneralProducto(models.Model):
    codigoProducto = models.CharField(max_length=50)
    nombreProducto = models.CharField(max_length=100)
    stock_actual = models.IntegerField()
    codigoNParte = models.CharField(max_length=50)
    #stock_restante =  (stock_maximo - stock_minimo)
    #nParteProducto = models.CharField(max_length=50)
# Create your models here.
class infoProductoBodega(models.Model):
    codigoBodega = models.CharField(max_length=50)
    codigoProductoPB = models.CharField(max_length=50)
    stockDisponible = models.IntegerField()
```



Luego, usando el archivo ***dbConnect.py,*** con todo lo antes mencionado, se procedió a guardar los registros en la base datos.

Ilustración 6: Insertando registros en la base PostGreSQL

```python
# INSERCION DE DATOS EN LA BASE DE DATOS POSTGRE SQL
for i in diccionario4:

    if (i in prodIdBD) and (i in diccionario4.keys()):

        sqlInstruction = infoGeneralProducto.objects.create(codigoProducto=diccionario1.get(i,i), nombreProducto= diccionario2.get(i,i),
                                                            stock_actual=diccionario3.get(i,i), codigoNParte=diccionario4.get(i, i))
```

El resultado la inserción de cada registro en la base de datos PostgreSQL se muestra en la ilustración 7:

Ilustración 7: registros cargados con éxito en la BD

Query Query History

```
1  select * from "gestionDB_infogeneralproducto"
2
```

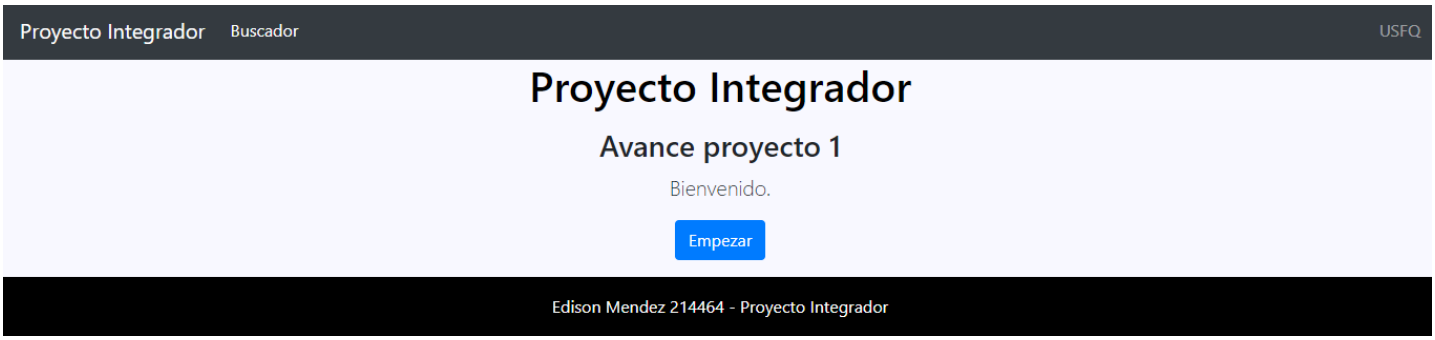
Data output Messages Notifications

	id [PK] bigint	codigoProducto character varying (50)	nombreProducto character varying (1000)	stock_actual character varying (100)	codigoNParte character varying (50)
1	568434	12050	COPETE/BOCIN/ENTRADA/EJE ...	4539.0	AX442
2	568435	2538	RULIMAN BOLITA LATON TRASM.	33375.0	AX303
3	568436	2224	HORNILLA PEQUENA 3 VUELTAS	187.0	LAXB510
4	568437	6775	CINTA MOMIA AIRE ACONDICIO	29643.0	AX308
5	568438	12490	FILTRO DE MANGUERA ROJO(E...	30550.0	AX306
6	568439	212	FILTRO LAVADORA MANGUERA	19327.0	AX317
7	568440	690	ANILLO ICE MAKER REFRIGERA	7322.0	AX377
8	568441	439	ACOPLE/CACHIMBA/PITON/PO...	6106.0	AX402

2.5. Aplicación web con Django

Para la parte del front-end, se diseñó una página web sencilla pero lo suficientemente clara como para mostrar los resultados que se espera.

Ilustración 8: Index web app



Se incluyó una parte de búsqueda en la cual, el vendedor podrá realizar sus búsquedas acordes a la información que requiera. La búsqueda es bidireccional, por tanto, el vendedor podrá realizar la búsqueda de la información del SKU por cualquier criterio de búsqueda, ya sea por: código, nombre o número de parte. Para este ejemplo, se realiza la búsqueda por código del producto, para así obtener toda la información que inicialmente se requería (especialmente el número de parte).

Ilustración 9: Realizando una búsqueda

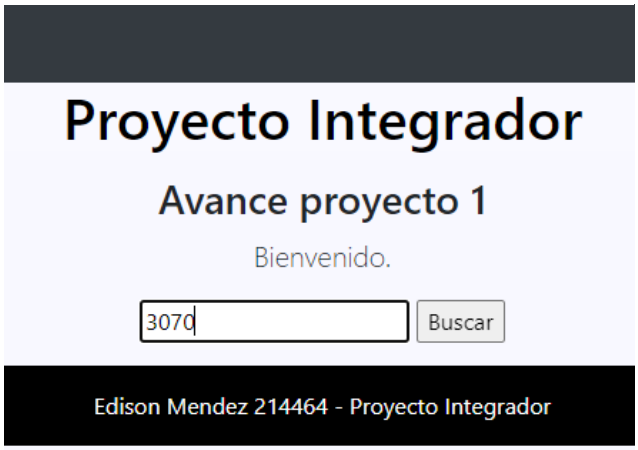


Ilustración 10: Resultados de la búsqueda

Una vez realizada la consulta, el buscador verificará los registros en la base de datos de PostgreSQL y los mostrará al vendedor. De este modo se puede observar el correcto funcionamiento de la aplicación web.

3. Ingeniería de datos y pronóstico de ventas históricas

Para esta segunda parte del proyecto integrador, se recopilaron datos **reales** de ventas históricas de la compañía donde trabaja el autor, comprendidas entre los años 2017 a 2022. Estos datos fueron proporcionados por el área de contabilidad de la compañía (reportes históricos generados en el ERP MBA3 en formato Excel), que muestran datos entre los más importantes: las ventas por producto, bodega y fecha de venta del producto.

3.1. Captación y procesamiento de los datos

Una de las cosas más importantes en una compañía siempre será la información. Con la información se puede realizar todo tipo análisis, desde los más simples hasta los más complejos. Por ejemplo, mediante el uso de Deep Learning, se puede crear una predicción de ventas usando datos históricos. Ahora bien, cuando se trata de resolver este tipo de problemas con el uso de inteligencia artificial, lo primero que se debe hacer es preguntar: ¿Qué es más importante, la cantidad o la calidad de los datos? El repositorio web especializado en analítica de datos, ***Decide***, dice que: *"En general, una mayor cantidad de datos conduce a modelos más fiables y por tanto mejores resultados, pero siempre que estos sean reales y representativos. Es preferible usar menos cantidad de datos, que más volumen, pero con una baja calidad"* (2019).

Para problemas que tienen que ver con series de tiempo (time series), la cantidad de características que se usen dependerá muchísimo del enfoque de análisis que se plantee dar. En este caso, se pretende realizar una predicción de la cantidad de ventas totales que la compañía realizó desde el año 2017 hasta el año 2022. Cada año tiene su propio documento de ventas, por lo que, para mayor facilidad de procesamiento de datos, se unieron todos estos archivos en uno solo. Este archivo consolidado, posee 8 características (columnas) con 1'048.576 instancias (filas). Las columnas en mención contienen la siguiente información:

- **Código del producto:** identificador único del código o SKU.
- Nombre del código
- **Fecha de venta del código:** fecha exacta de venta del producto. Esta columna será de suma importancia en la predicción.
- **Bodega que realizó la venta:** identificador único de la bodega (tienda/sucursal) que realizó la venta.
- **Nombre del cliente que realizó la compra:** nombres completos del cliente.
- **Unidades vendidas:** cantidad que el cliente compró.
- **Precio:** precio al que se le vendió, este precio suele ser variable dependiendo el cliente.
- **Venta total (dólares):** multiplicación de las unidades vendidas por el precio. Dicho de otra manera, la ganancia total en dólares que se obtuvo por la venta de dicho producto a dicho cliente.

Dependiendo del enfoque del problema que se quiera analizar, mucha de esta información puede ser redundante o poco útil. Como se explicó anteriormente, el problema que se pretende resolver es la predicción de ventas totales por año que genere la empresa. En series de tiempo, es sumamente importante la columna de fechas, ya que esta columna será el rótulo para poder realizar análisis día a día, mes a mes, o año a año. En este caso, el enfoque que se requiere son las ventas totales de las unidades vendidas que se realizaron **mes a mes**, en el período de tiempo del 2017 al 2022. Por tal razón, las únicas columnas que se usarán para este análisis serán: la columna fecha y la columna unidades vendidas. En series de tiempo, esto corresponde a un problema de tipo **univariado,** debido a que se usa una única columna para el análisis predictivo.

3.2. Análisis exploratorio de datos (EDA)

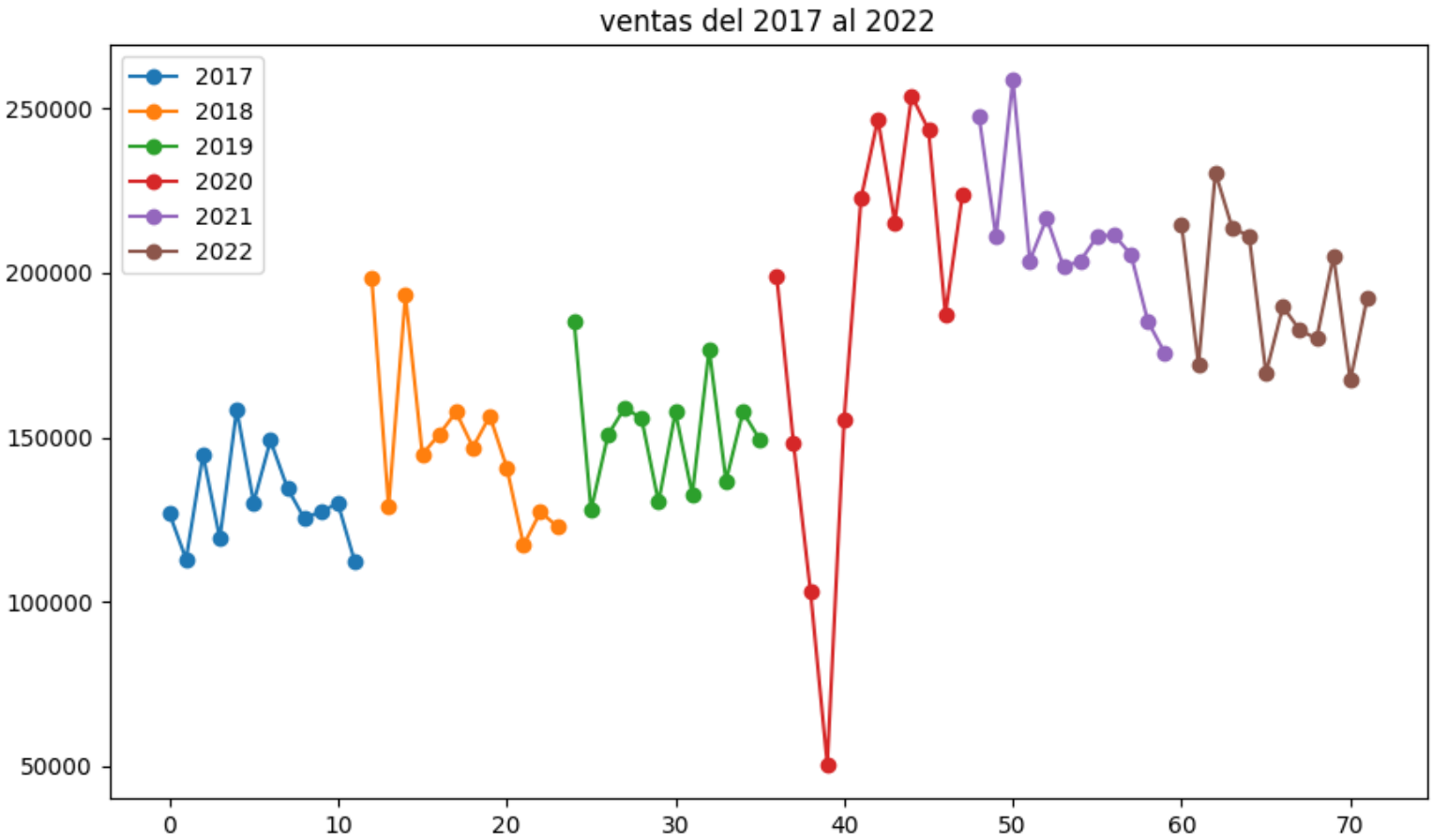
3.2.1. Reducción de la numerosidad

Como se explicó anteriormente, el análisis que se realizará será mes a mes. Para ello, primero se tiene que hacer un estudio correcto de los datos, a esta técnica se la conoce como **análisis exploratorio de datos**. Esta técnica permite observar qué tanta variabilidad tiene las ventas mes a mes en el período de tiempo mencionado. Para ello se agruparon las ventas totales mes a mes del dataframe consolidado indicado anteriormente; dando como resultado un nuevo dataframe con el que se iniciará el análisis.

```
date,Unidades vendidas
2017-01-31,127037.25
2017-02-28,112951.5
2017-03-31,144805.6
2017-04-30,119257.5
2017-05-31,158511.82
2017-06-30,130230.1
2017-07-31,149060.5
2017-08-31,134656.4
2017-09-30,125392.2
2017-10-31,127508.61
2017-11-30,129913.05
2017-12-31,112143.55
2018-01-31,198318.32
2018-02-28,128962.95
```

Este dataframe posee solamente 2 columnas con 72 filas cada una, representan los 12 meses en el período de tiempo de 6 años (2017 a 2022), lo que hace que la serie de tiempo sea de tipo **univariada,** tal como se explicó anteriormente. Para series de tiempo, la columna **date** siempre tendrá que ser **datetime,** y el índice del dataframe para un correcto análisis. Para una mejor visualización de los datos, en la ilustración 11, se puede ver la distribución de las unidades vendidas en el período de tiempo antes mencionado:

Ilustración 11: análisis exploratorio de datos: ventas desde 2017 a 2022



Como se observa, para el período del año 2020, las ventas en el mes de abril son demasiado bajas, en comparación con los demás años; esto se debe en gran parte a que precisamente en ese año y en ese mes comenzó la pandemia por covid-19 y el confinamiento a todos los ciudadanos en sus casas. No obstante, este año también será parte del análisis.

3.3. Predicción usando series de tiempo LSTM

3.3.1. Series de tiempo y sus características

Los problemas que incluyen análisis de datos durante intervalos de tiempo se conocen como **series de tiempo (time series).** Este tipo de problemas tiene 2 características principales:

- **Tendencia:** Se refiere a la variación de valores en un período de tiempo. Cuando los valores crecen al pasar el tiempo, se considera serie *alcista*, cuando decrece, *bajista*, cuando es constante, *estacionaria* (Aigiomawu, 2018).
- **Tener estacionalidad:** Es decir, debe tener un comportamiento similar mes a mes, año a año. Dicho de otra forma, una serie de tiempo estacionaria se debe mantener constante en media y desviación estándar a través del tiempo.

Para determinar si una serie de tiempo es estacionaria o no, se la puede determinar de dos formas:

- **1: Analizando la gráfica, su media y su desviación estándar**. De la *ilustración 11*, se puede observar que, para este caso, la serie es **no** estacionaria, ya que se puede ver que las ventas crecen o decrecen muy rápido a lo largo de los años. Es decir, la gráfica indica que con el pasar del tiempo, más se va a vender. Por tal razón se deben convertir los datos en estacionarios.
- **2**: **Dickey-Fuller Test:** este test permite determinar numéricamente si una serie es estacionaria o no; el test se basa en la hipótesis nula, que indica que toda serie es no

estacionaria. Para ello se incluye estadísticas de prueba y valores críticos. Para rechazar la hipótesis nula, e indicar que una serie es estacionaria, se debe demostrar que las ***estadísticas de prueba*** sean menores que los ***valores críticos y que el p-value sea menor que 0.05.*** (Jain, 2016). Aplicando este test mes a mes, a las unidades vendidas por cada año, se obtuvieron los siguientes resultados:

Ilustración 12:primer prueba de estacionalidad: Dickey-Fuller test

```
Results of Dickey-Fuller Test:
Test Statistic                -2.705356
p-value                        0.073116
#Lags Used                     1.000000
Number of Observations Used   70.000000
Critical Value (1%)           -3.527426
Critical Value (5%)           -2.903811
Critical Value (10%)          -2.589320
```

Como se puede ver, para 2 de los 3 valores críticos el valor de test es mayor, a su vez el p-value es mayor que 0.05, por lo que se confirma que las series son **no estacionarias.**

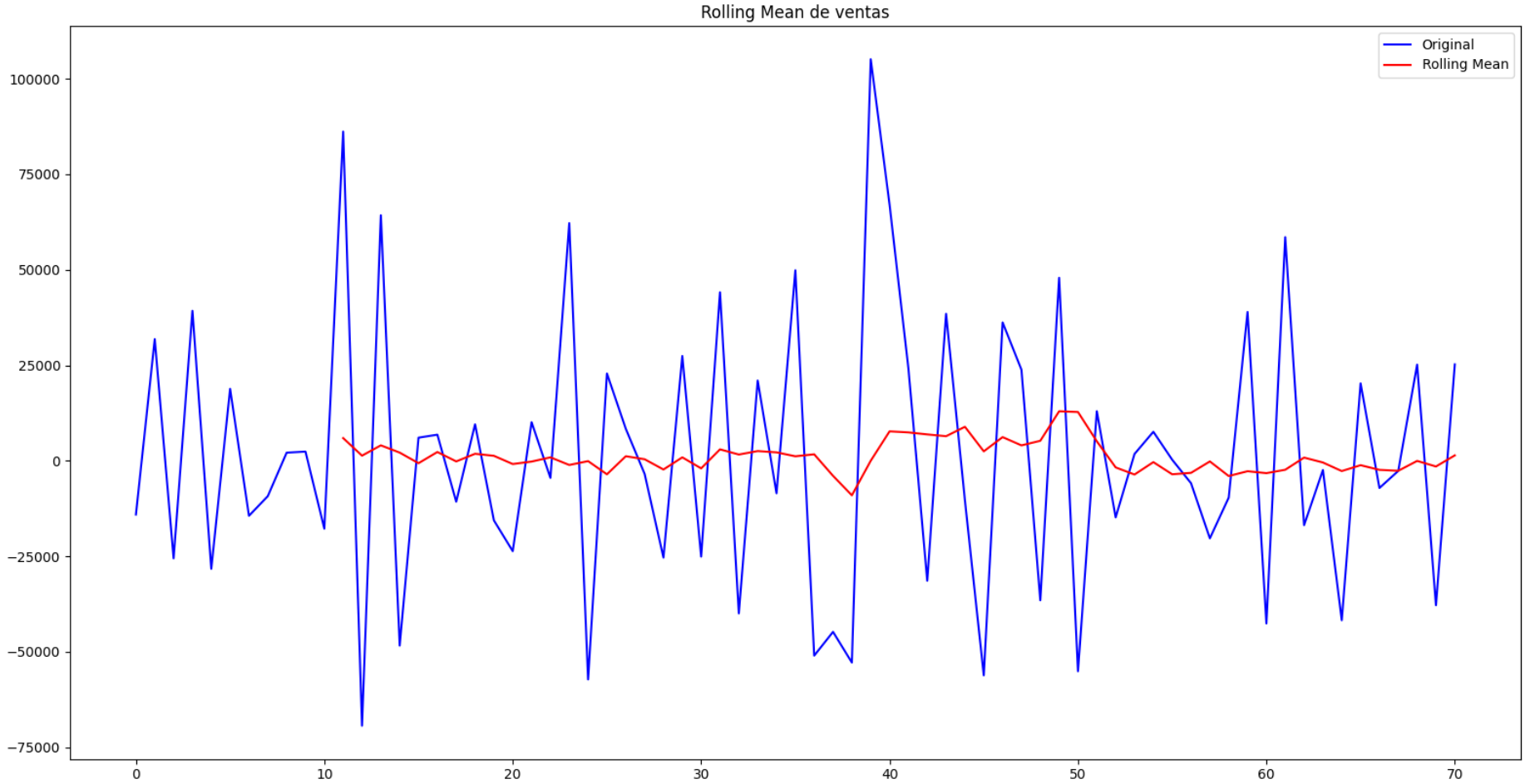
3.3.2. Conversión de datos en estacionarios

En la realidad es casi imposible asegurar que muchos de los modelos de time series son completamente estacionarios, pero para resolver este tipo de problemas es necesario hacerlo lo más aproximado posible. Para obtener una serie estacionaria es necesario estimar y eliminar las tendencias. Como se observa en la ilustración 11, se puede ver que las ventas crecen conforme pasan en el tiempo (a excepción del mes abril en 2020 que decrecen bruscamente).

Para eliminar la tendencia en los datos, se pueden utilizar muchísimas técnicas de reducción, que harán que los datos se transformen temporalmente para que sean usados en predicciones con series de tiempo. El objetivo de estas técnicas es deshacerse de la media variable en los datos. Para este caso, se usó la técnica de diferenciación; que calcula la diferencia de los términos consecutivos, logrando que la media variable de los datos se centre en el 0 lo máximo posible; es decir, logrando que la media **no** varíe tanto con respecto al

tiempo. Una vez aplicada esta técnica se consiguen los resultados mostrados en la ilustración 13:

Ilustración 13: Técnica de diferenciación en los datos para converir la serie en estacionaria



Como se puede ver, la media variable (dibujada en color rojo) logra una mayor estabilidad, haciendo que a simple vista se considere que la gráfica sí es estacionaria. Para corroborar lo antes dicho, se evalúa la estacionalidad una vez más con el test de Dickey-Fuller:

Ilustración 14:segunda prueba de estacionalidad: Dickey-Fuller test

```
Results of Dickey-Fuller Test:
Test Statistic                -6.832347e+00
p-value                        1.879718e-09
#Lags Used                     2.000000e+00
Number of Observations Used    6.800000e+01
Critical Value (1%)           -3.530399e+00
Critical Value (5%)           -2.905087e+00
Critical Value (10%)          -2.590001e+00
dtype: float64
```

En los resultados, la estadística de test se observa que resultó ser mucho menor que los 3 valores críticos; y que el p-value tiene un valor $1{,}89^{-9}$, lo cual es muchísimo menor que 0,05; por lo que con toda seguridad se puede decir que se trata de una serie estacionaria.

3.3.3. Feature Engineering para series de tiempo

3.3.3.1. Convertir serie temporal en un problema de aprendizaje supervisado

Los problemas de aprendizaje supervisado se componen de patrones de entrada/inputs (x), y patrones de salida/outputs (y), de modo que un algoritmo puede predecir una salida por medio de los patrones de entrada. Para poder usar una red neuronal LSTM, o cualquier red neuronal que sea de tipo supervisada, es necesario contar con la suficiente cantidad de datos que permitan entrenar a la red correctamente.

Una de las técnicas más usadas para convertir un problema de series de tiempo, en un problema de aprendizaje supervisado, es usar la técnica de **distribuited lags.** Esta técnica no hace más que crear copias de las filas y desplazarlas, de modo que hay una relación directa

entre la columna actual y la siguiente, que en este caso es t+1 (Brownlee, 2019). Como el problema se centra en predecir la venta de los próximos 12 meses, se crean 12 columnas bajo este patrón. A continuación, se muestra un ejemplo de la aplicación de esta técnica:

Ilustración 15: funcionalidad de la técnica distributed lag

		t	t-1
1	0	0	NaN
2	1	1	0.0
3	2	2	1.0
4	3	3	2.0
5	4	4	3.0
6	5	5	4.0
7	6	6	5.0
8	7	7	6.0
9	8	8	7.0
10	9	9	8.0

Como se puede ver, la columna *t-1*, es un desplazamiento t-1 de la columna *t.* Es necesario mencionar que cuando se aplica esta técnica, se crean valores *nan* o nulos en la primera columna del desplazamiento. Estos valores tendrán que ser eliminados. Bajo este mismo concepto se aplicó esta técnica para este problema de series de tiempo, en donde la columna a ser 'copiada' será la columna de unidades vendidas. Por tanto, cada fila será un mes y cada columna será una representación de ventas. Este nuevo dataset es el que se usará para poder entrenar el modelo.

Ilustración 16: dataset supervisado aplicado la técnica distributed lag

date	Unidades vend	unidadesVen	t-0	t-1	t-2	t-3	t-4	t-5	t-6	t-7	t-8	t-9	t-10	t-11	t-12
28/2/2018	128962.95	##########	128962.95	198318.32	112143.55	129913.05	127508.61	125392.2	134656.4	149060.5	130230.1	158511.82	119257.5	144805.6	112951.5
31/3/2018	193228.65	64265.7	193228.65	128962.95	198318.32	112143.55	129913.05	127508.61	125392.2	134656.4	149060.5	130230.1	158511.82	119257.5	144805.6
30/4/2018	144842.45	##########	144842.45	193228.65	128962.95	198318.32	112143.55	129913.05	127508.61	125392.2	134656.4	149060.5	130230.1	158511.82	119257.5
31/5/2018	150885.6	##########	150885.6	144842.45	193228.65	128962.95	198318.32	112143.55	129913.05	127508.61	125392.2	134656.4	149060.5	130230.1	158511.82
30/6/2018	157715.95	##########	157715.95	150885.6	144842.45	193228.65	128962.95	198318.32	112143.55	129913.05	127508.61	125392.2	134656.4	149060.5	130230.1
31/7/2018	146994.35	##########	146994.35	157715.95	150885.6	144842.45	193228.65	128962.95	198318.32	112143.55	129913.05	127508.61	125392.2	134656.4	149060.5
31/8/2018	156536.55	##########	156536.55	146994.35	157715.95	150885.6	144842.45	193228.65	128962.95	198318.32	112143.55	129913.05	127508.61	125392.2	134656.4
30/9/2018	140931.65	##########	140931.65	156536.55	146994.35	157715.95	150885.6	144842.45	193228.65	128962.95	198318.32	112143.55	129913.05	127508.61	125392.2
31/10/2018	117268.13	##########	117268.13	140931.65	156536.55	146994.35	157715.95	150885.6	144842.45	193228.65	128962.95	198318.32	112143.55	129913.05	127508.61
30/11/2018	127367.58	##########	127367.58	117268.13	140931.65	156536.55	146994.35	157715.95	150885.6	144842.45	193228.65	128962.95	198318.32	112143.55	129913.05
31/12/2018	122883.2	##########	122883.2	127367.58	117268.13	140931.65	156536.55	146994.35	157715.95	150885.6	144842.45	193228.65	128962.95	198318.32	112143.55
31/1/2019	185099.6	##########	185099.6	122883.2	127367.58	117268.13	140931.65	156536.55	146994.35	157715.95	150885.6	144842.45	193228.65	128962.95	198318.32

3.3.4. Creación del modelo

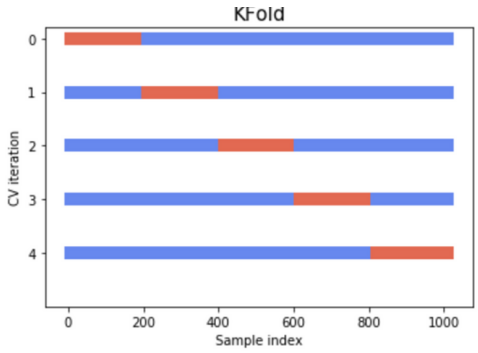
3.3.4.1. División del dataset para test y train

3.3.4.1.2 Diferencias entre K-fold-cross-validation y TimeSeriesSplit

En aprendizaje supervisado, para validar la trazabilidad de un modelo se usan técnicas de validación, entre las más conocidas están **K-fold-cross-validation** y **TimeSeriesSplit.** Estas dos técnicas lo que hacen es evaluar y dividir un dataset en partes para el train y test del modelo.

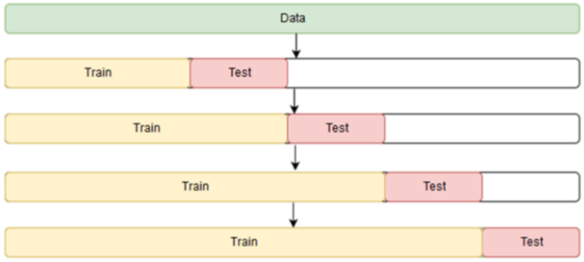
Para el caso de la técnica **K-fold-cross-validation,** lo que hace es dividir en *folds* de forma aleatoria el dataset para train y test, dado un *k* que representa el número de divisiones. Al ser aleatoria, esta técnica no es secuencial, por lo que para series de tiempo no funciona, ya que usa el futuro para poder predecir el pasado, lo cual no tiene sentido.

Ilustración 17: ***K-fold-cross-validation***



Por otra parte, la técnica Time Series Split, sí divide de forma secuencial el dataset dado un *k* que representan las divisiones. Esta técnica sí resulta útil ya que utiliza el pasado para poder predecir el futuro, que es lo que se plantea hacer en este problema de series de tiempo.

Ilustración 18: Time Series Split



Por tal razón, bajo el concepto de TimeSeriesSplit se dividió el dataset, de modo que, para el train, se usaron todos las filas y columnas desde 2018 hasta el 2021, y para el test, se usó el año 2022. De modo que el tamaño del test y train quedó de esta forma:

Ilustración 19: tamaño del test y train

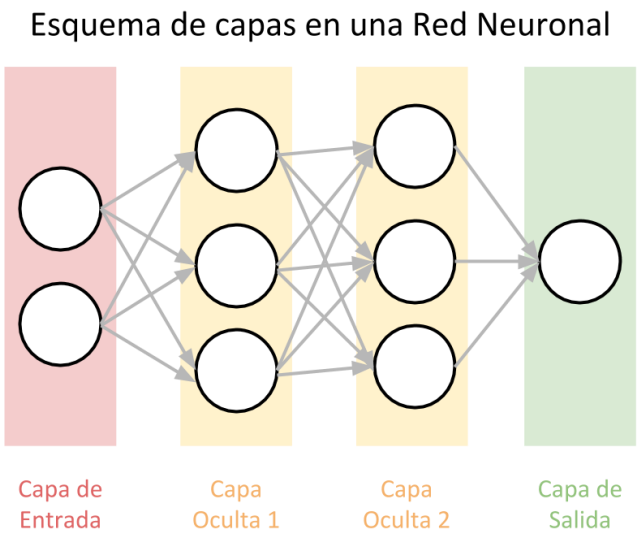
```
Shape of X Train: (47, 13)
Shape of y Train: (47,)
Shape of X Test: (12, 13)
Shape of y Test: (12,)
```

Es importante mencionar que, las columnas que se tomaron en cuenta para el train y test fueron las columnas *sales_diff* y los *lags (las 12 copias de t-i de las ventas)*, es por ese motivo que aparecen 13 columnas en las tuplas para cada una.

3.3.4.2. Características de una red neuronal fully-connected

La estructura básica de una red neuronal fully-connected es la siguiente:

Ilustración 20: esquema de una red neuronal fully-connected



Capa de entrada: se define el tipo de red neuronal que se usará (ej.: LSTM), el número de neuronas que tendrá la capa.

Capas ocultas: se pueden definir ***n*** capas ocultas en la red neuronal, con ***n*** neuronas, esto dependerá mucho del tipo de problema que se esté resolviendo. Las capas ocultas siempre serán las que estén en medio de la capa de entra y la capa de salida.

Capa de salida: la capa de salida puede tener ***n*** neuronas y dependiendo el enfoque, esta capa puede ser usada para modelos de clasificación o regresión.

Para el problema que se quiere resolver, se usarán redes neuronales de tipo LSTM; estas redes son de tipo supervisado, que pueden ser usadas en problemas de clasificación o predicción, siendo los problemas de predicción en series de tiempo un tipo de problemas muy adecuados a este tipo de red. Por esta razón se usó una red neuronal LSTM.

3.3.4.3. Construcción del modelo

En el aprendizaje profundo, la gran mayoría de arquitecturas pueden ser construidas a partir de combinaciones de algunos tipos de capas (ej.: fully-connected). En este caso se usa la librería Keras de Python, en donde las redes neuronales pasan a ser combinaciones de capas. Aquí se usa una modelo de red *Sequential()* que permite que las capas subsiguientes tengan un conocimiento jerarquizado. Es decir, cada capa, recibirá información de las capas anteriores y estas capas, enviarán información a las siguientes capas con los cálculos de las sumas ponderadas por neurona. Entre más capas se añadan, más compleja será la arquitectura que se elabore. Sin embargo, es importante entender que esto no garantiza que a mayor cantidad capas añadidas, con mayor número de neuronas, los resultados siempre serán mejores. Esto se debe a que los modelos demasiado complejos pueden provocar un sobreajuste (overfitting) y también un mayor uso de los recursos del ordenador; lo que hará que el modelo no sea eficiente. Por tal razón es importante encontrar un equilibrio en la complejidad de la tarea que se planea resolver.

3.3.4.3.1. Qué es una neurona y sus características

Cuando se crea una red neuronal se debe definir el número de capas de la red (estas capas pueden ser de entrada, salida, o capas ocultas), el número de neuronas que llevará la capa, y el tipo de función de activación de la capa. Una neurona, también conocida como perceptrón es la unidad principal dentro de una red neuronal. Las neuronas tienen estas características:

- **Entrada:** Reciben valores de entrada de otras neuronas o de los datos de entrada. Cada entrada está asociada con un peso que representa la importancia de la esa entrada para la neurona.
- **Suma ponderada:** calcula la suma ponderada de sus entradas multiplicando cada entrada por su peso correspondiente, y luego las suma.
- **Función de activación:** esta suma ponderada se pasa a través de una función de activación para evitar que la suma ponderada de todas las neuronas provoque una salida lineal. La función de activación toma las sumas ponderadas y añade deformaciones no-lineales, para que así se puedan concatenar de forma efectiva la computación de varias neuronas. Estas deformaciones dependen de la función de activación que se usa. La importancia de usar una función de activación radica en que, sin una función de activación, cada neurona realizará su suma ponderada y sin importar la profundidad de la red, todas colapsarán en una sola neurona.
- **Salida:** el resultado de la función de activación se convierte en la salida de la neurona, que, dependiendo de la arquitectura, se pasará a otras neuronas como entradas.

En la arquitectura de la red neuronal se usó una función de activación ***tanh,*** esta función de activación provoca una distorsión haciendo que los valores más grandes se saturen en 1, y

los valores más pequeños en -1, por tanto, su rango es [-1,1]. Gráficamente esta función se comporta así:

Ilustración 21: comportamiento gráfico de la función de activación tanh

3.3.4.4 Arquitectura de la red neuronal

Normalización de datos con MinMax-Scaler: Se creó un método para la normalización de los datos, esta técnica es importante para llevar los datos a un rango [-1,1]. Estos mismos datos son reversados a su estado original después para poder evaluar el modelo una vez que se haya entrenado la red correctamente. Con la ayuda de la clase Keras, para la arquitectura de la red neuronal se creó una red con las siguientes características:

- **Capa de entrada**: una capa de entrada LSTM con 4 neuronas. Se define también un batch_input_shape=(1,1,13) para establecer la forma de los datos de entrada. Siendo batch_size = 1, timesteps = 1, e input_dim=13.
- **Capa oculta**: una capa oculta *dense* con tres neuronas con una función de activación ***tanh***.
- **Capa de salida**: una capa de salida *dense* sin función de activación (esto se debe a que, en problemas de predicción, la capa de salida calcula directamente el valor predicho).

Gráficamente, la red neuronal tiene esta estructura:

Ilustración 22: arquitectura de la red neuronal LSTM configurada

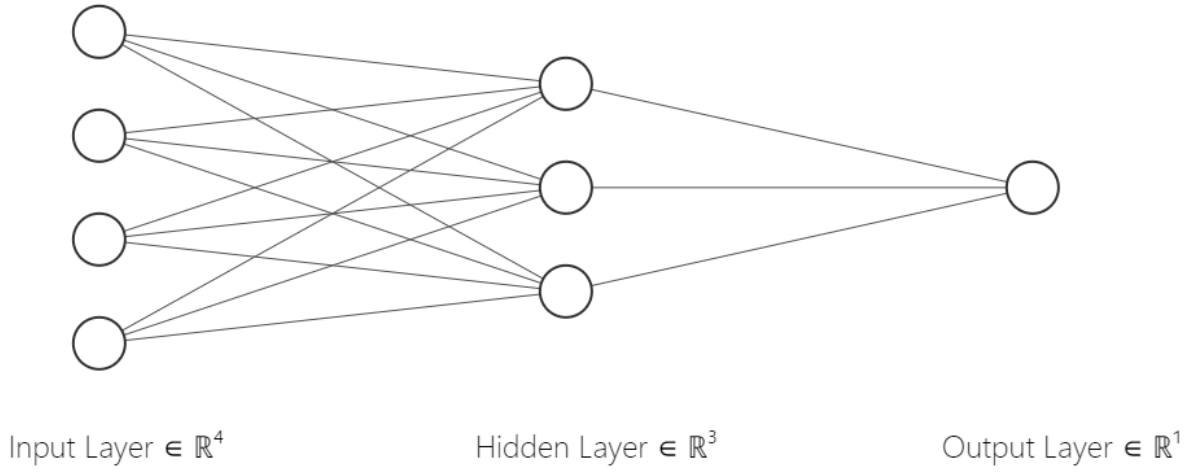


Ilustración 23: creación del modelo sequential() de keras para crear la red neuronal en Python

```
model = Sequential()
model.add(LSTM(4, batch_input_shape=(1, X_train.shape[1], X_train.shape[2]),stateful=True))
model.add(Dense(3, activation='tanh')) #capa oculta
model.add(Dense(1)) #capa de salida
```

En resumen, se creó la red neuronal con 3 capas y 8 neuronas en total.

3.3.4.4.1. Entrenamiento de la red neuronal

Para alcanzar un resultado satisfactorio en un problema de aprendizaje supervisado, no basta con preocuparse sólo por la arquitectura, sino también por los hiperparámetros de la red neuronal. “Entonces, lo que **un optimizador** hace es, obviamente, **optimizar los valores de los parámetros para reducir el error cometido por la red**. El proceso mediante el cual se hace esto se conoce como “backpropagation” (Martínez, 2020)”. En resumen, un optimizador es un algoritmo que permite minimizar una función de pérdida de un modelo predictivo en un conjunto de datos de entrenamiento. Con el uso de la librería Keras, el entrenamiento de la red se realiza compilando el modelo, para que así se comunique al back-end de tensorflow asegurando que la red está lista para entrenar. En esta configuración se especifica qué función de costo, y qué optimizador utilizaré para entrenar la red neuronal. En la configuración del optimizador, se usó como métrica de loss, el mean squared error (mse), y como métrica el

'accuracy'. Finalmente, como optimizador, se usó **el descenso del gradiente estocástico, sgd**; este es un método básico para optimización de redes neuronales y muy eficaz en algoritmos de aprendizaje profundo.

Ilustración 24: compilación de la red, con optimizador sgd

```
model.compile(loss='mse', optimizer='sgd', metrics=['accuracy'])
```

Una vez compilado el modelo, simplemente se debe entrenar la red con la función *.fit()* de la clase Keras. Los datos de entrada de la red estarán dados por *X_train*, los datos de salida por *y_train*, y se realizará una validación con los datos de *X_test* y *y_test* para medir el *loss* del entrenamiento. También se realiza el entrenamiento para 200 épocas, en este lapso los resultados fueron óptimos ya que no hubo sobreajuste en los resultados.

Ilustración 25: inicio de entrenamiento de la red neuronal

```
history = model.fit(X_train, y_train, validation_data=(X_test,y_test), epochs=200,
                    batch_size=1, verbose=1,shuffle=False)
```

Pasadas las 200 épocas se pudo ver problemas de overfitting gráficamente, por lo que ya no fue necesario entrenar el modelo a una mayor cantidad de épocas. A continuación, se muestran más detalles de la arquitectura de la red neuronal:

Ilustración 26: información más detallada del entrenamiento de la red

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (1, 4)                    288

 dense (Dense)               (1, 3)                    15

 dense_1 (Dense)             (1, 1)                    4

=================================================================
Total params: 307
Trainable params: 307
Non-trainable params: 0
```

En la ilustración 26, la columna ‘param’ indica el número de parámetros (pesos y sesgos) asociados a cada capa; estos parámetros se ajustan acorde el algoritmo de optimización descenso del gradiente estocástico (sgd). Dicho de otra manera, los parámetros entrenables representan al número total de parámetros que se pueden aprender en el modelo; esta métrica es importante ya que afecta directamente a la complejidad y capacidad del modelo. La capa de entrada LSTM, posee 288 parámetros; la capa oculta dense con 3 neuronas, posee 15 parámetros; y la capa de salida dense con una neurona posee 4 parámetros. En total, los parámetros entrenables bajo esta configuración son 307.

Una vez que la red se terminó de entrenar, la red ya estará capacitada para poder arrojar resultados. Como en el entrenamiento se realizó una validación con el conjunto de test, para ‘predecir’ el resultado de la red neuronal, se pasa como parámetro este mismo conjunto de datos. La configuración se la hace de esta manera:

Ilustración 27: obtener las predicciones usando predict de Keras

```
predictions = model.predict(X_test, batch_size=1)
```

Llegado a este punto es importante mencionar que, lo que se espera del objeto *predictions* es un resultado de la predicción de las ventas de los próximos 12 meses que la compañía realizará. Esto se debe a que el conjunto de *test* tiene las ventas de los últimos 12 meses del dataset, es decir, el año 2022. Por tal, la comparación de los resultados ‘predichos’ se realiza con las ventas realizadas en este último año, es decir, con el conjunto de test. Si se muestra los resultados de este objeto se esperaría obtener un arreglo 2D, con l2 filas con datos normalizados (ya que anteriormente se normalizaron los datos en el rango [-1,1] con la función MinMax-Scaler). A continuación, se muestran dichos resultados:

Ilustración 28: resultados de la predicción en formato normalizado

```
predictions LSTM:
[[ 0.22604866]
 [-0.40119514]
 [ 0.2945646 ]
 [-0.3145226 ]
 [-0.21849841]
 [-0.6731553 ]
 [-0.03069832]
 [-0.16738318]
 [-0.26038387]
 [-0.05782285]
 [-0.6670035 ]
 [-0.0962755 ]]
```

Ahora estos datos deben ser llevados a su espacio original para tener resultados que se puedan interpretar y comparar. Aplicando el método *reverseProcessing()* se pasa como argumento esta matriz y se hace una transformación inversa de estos datos. Para visualizar los resultados de la predicción, se creó el método $predictDf(unscaledPred, df\_mes)$: este método fue creado con el fin de generar los resultados una vez que la red LSTM sea evaluada. En este método se exportan los resultados en un dataframe para poder visualizar de mejor manera, con qué precisión el modelo pudo predecir la cantidad de ventas. A continuación, se muestra un ejemplo con los resultados obtenidos durante la predicción:

Ilustración 29: resultados de la predicción

	date	org_value	pred_value	diff[org-pred]	%error
0	2022-01-31	175647.0	196776	-21129.0	12.029
1	2022-02-28	214606.0	183157	31449.0	14.654
2	2022-03-31	172014.0	222652	-50638.0	29.438
3	2022-04-30	230548.0	229117	1431.0	0.621
4	2022-05-31	213678.0	205080	8598.0	4.024
5	2022-06-30	211260.0	174115	37145.0	17.583
6	2022-07-31	169510.0	189160	-19650.0	11.592
7	2022-08-31	189770.0	187432	2338.0	1.232
8	2022-09-30	182645.0	172251	10394.0	5.691
9	2022-10-31	179996.0	215543	-35547.0	19.749
10	2022-11-30	205159.0	179191	25968.0	12.657
11	2022-12-31	167330.0	192367	-25037.0	14.963

Adicional a todo lo antes mencionado, también se realizó lo siguiente:

Calificar el modelo en el método $scoreModel(unscaledData, dfMesoriginal)$: en este método se evalúa el performance que tuvo la red neuronal en la fase de entrenamiento. Las métricas que se usaron para la red fueron las siguientes:

- $\boldsymbol{RMSE}$: error cuadrático medio, mide la cantidad de error entre dos conjuntos de datos.
- $\boldsymbol{MAE}$: Es la diferencia entre el valor pronosticado y el valor real en cada punto pronosticado.
- $\boldsymbol{R^2}$**:** Es el coeficiente de determinación que mide la capacidad de un modelo para predecir futuros resultados, los valores de la métrica varían entre 1-0, siendo 1 el mejor resultado posible medido. Esto ocurre cuando la predicción coincide con los valores de la variable objetivo.
- $\boldsymbol{MAPE}$**:** Error porcentual medio absoluto se utiliza para medir la precisión de un modelo, cuanto menor sea el valor de MAPE, mejor será el modelo para predecir resultados.

3.3.4.5. Evaluación del modelo y muestra de resultados

Como se había explicado anteriormente, el objetivo es poder predecir las ventas que se realizarán en un futuro próximo. Para ello, se usaron los datos de las ventas entre el 2017 al 2022, siendo las ventas desde 2017 a 2021 como datos de training, y las ventas del 2022 como test. Por tal razón y para ver los resultados, gráfica y numéricamente de la precisión de la predicción realizada, se colocó encima del 2022 los resultados que serían predichos para el '2023'. A continuación, se muestran resultados:

Ilustración 30: resultados numéricos detallados de la mejor predicción y scores del modelo

	date	org_value	pred_value	diff[org-pred]	%error
0	2022-01-31	175647.0	211730	-36083.0	20.543
1	2022-02-28	214606.0	174907	39699.0	18.499
2	2022-03-31	172014.0	220658	-48644.0	28.279
3	2022-04-30	230548.0	206043	24505.0	10.629
4	2022-05-31	213678.0	207976	5702.0	2.669
5	2022-06-30	211260.0	174029	37231.0	17.623
6	2022-07-31	169510.0	188769	-19259.0	11.362
7	2022-08-31	189770.0	179895	9875.0	5.204
8	2022-09-30	182645.0	185643	-2998.0	1.641
9	2022-10-31	179996.0	199364	-19368.0	10.760
10	2022-11-30	205159.0	163283	41876.0	20.411
11	2022-12-31	167330.0	185703	-18373.0	10.980

RMSE: 5334.157477990315
MAE: 4766.5
R2: 0.9274977832441469
MAPE: 0.024234911779075833

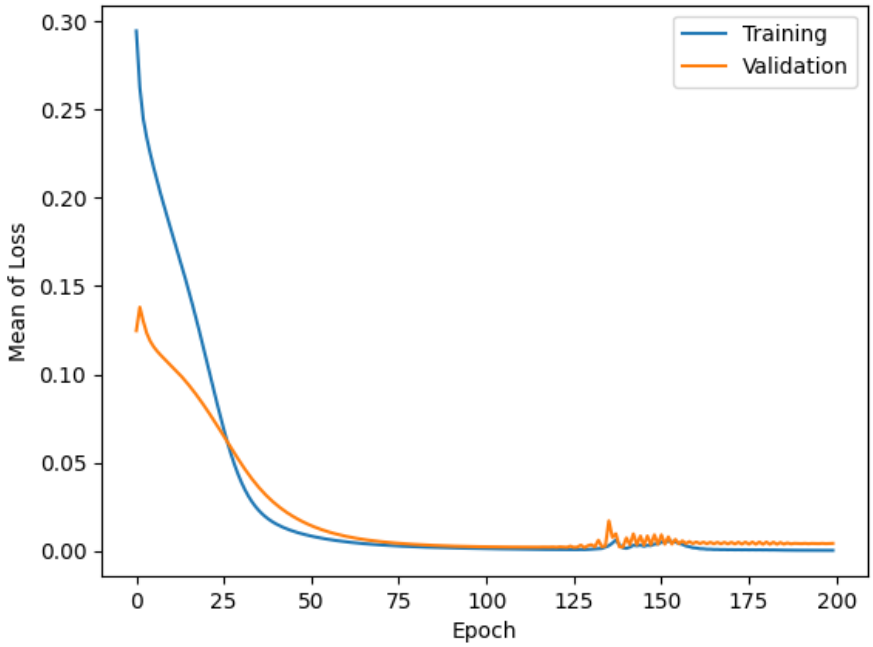
Como se puede ver, en los resultados de la columna pronosticada, **pred_value,** en contraste con la columna **org_value,** la diferencia no resulta ser tan grande. Por lo que el porcentaje de error no resulta ser tan alto. Incluso, en algunos casos, el porcentaje de error llega a ser mucho menor que el 10%. Ahora evaluando el desempeño del modelo con las métricas, se puede ver que se obtuvo una buena puntuación en la métrica R2, por lo que se puede asegurar que el modelo tuvo la capacidad de poder predecir en un 92% de forma correcta. También se puede corroborar lo expuesto antes, viendo la puntuación de la métrica MAPE, que indica que a menor que se el valor medido, mejor será el desempeño del modelo para predecir.

3.3.4.6 Plot Loss epoch vs validation

La gráfica de loss vs epoch, permite entender el comportamiento que tuvo el modelo en el entrenamiento. Como se puede ver, ambas curvas empiezan a converger a partir de la época 50, hasta la época 130 aproximadamente. A partir de ahí se observa un overfitting desde esa

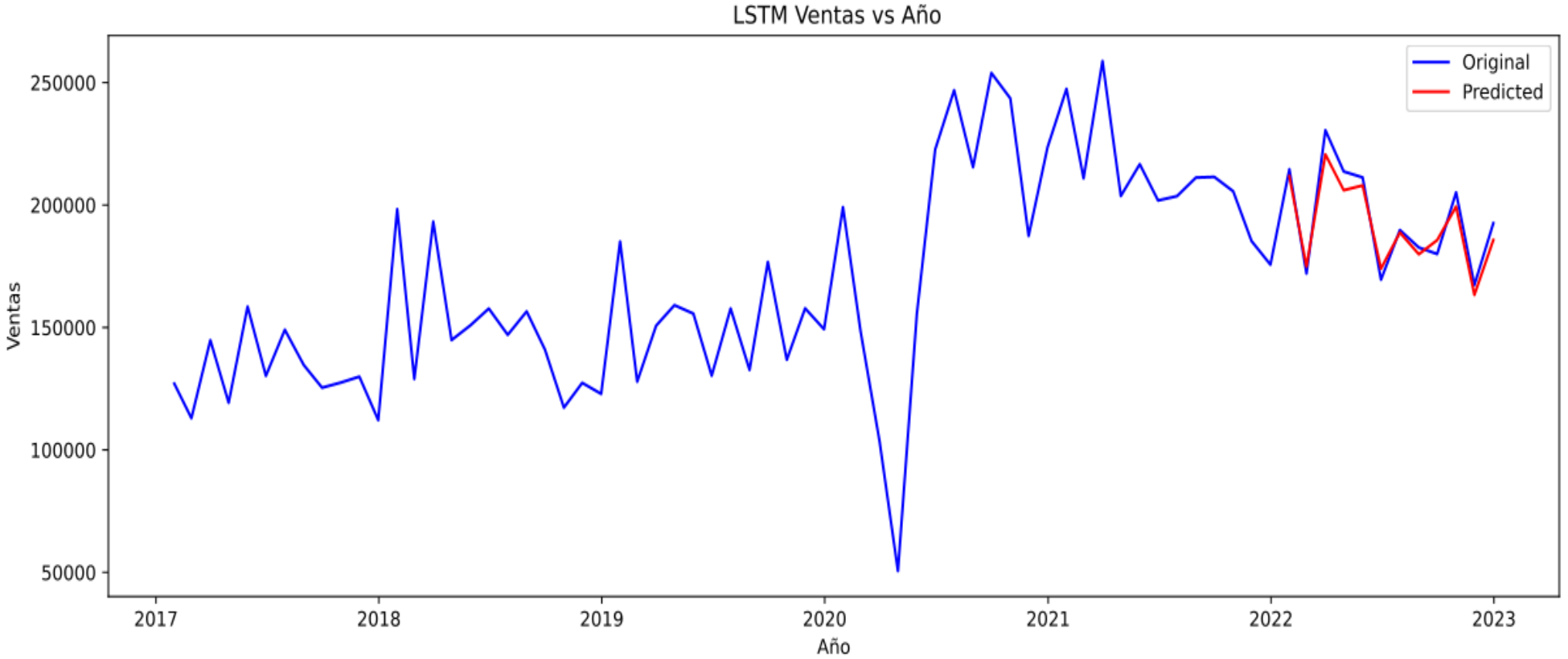
época, hasta la época 150, y continúa en overfitting ligero hasta la época 200. A pesar de esto, la red tuvo la capacidad de predecir de forma correcta, por lo que se concluye que el overfitting en este caso no fue determinante en la obtención de los resultados.

Ilustración 31: gráfica Loss vs Epoch del training y validation



3.3.4.7. Plot de resultado de predicción de ventas

Ilustración 32: resultado final gráfico del pronóstico de ventas



Finalmente, analizando los resultados de la predicción gráficamente en la ilustración anterior, se puede observar que la red pudo predecir bastante bien, alcanzando una curva bastante similar a las ventas generadas en el 2022. Ahora bien, en la práctica para la obtención de estos resultados mostrados, se tuvo que realizar varias pruebas para poder obtener los mejores resultados. Entre ellas estuvo el configurar bien la arquitectura de la red neuronal LSTM, enfocando las métricas aplicables a series de tiempo. Para este caso, en particular de aprendizaje profundo, se pudo determinar que el mejor optimizador para predicciones con redes neuronales es el optimizador Adam. También, la función de activación tangente hiperbólica (tanh), fue superior a funciones de activación tales como sigmoide o relu. Por tal razón, se concluye que la predicción de ventas fue eficaz, tanto gráfica, numérica y estadísticamente.

CONCLUSIONES

La correcta interacción entre el back end y el front end de la aplicación web desarrollada en Django, se debe en gran medida a la correcta comunicación que se estableció con la base de datos SQL Server del sistema MBA3, con la base de datos en Excel. Para relacionar ambas tablas, se usó un diccionario en el que la clave primaria fue precisamente un el código SKU de un producto, ya que, al ser un identificador único, fue fácil identificar cada producto y así captar toda su información que posteriormente fue almacenada en una nueva base de datos, PostgreSQL. Por tal, desde esta base de datos se obtiene la información que se muestra en el front-end de la aplicación. El vendedor ahora únicamente necesita hacer su búsqueda por código o nombre, para así poder visualizar no solo el código que busca, sino productos relacionados a ese código en la aplicación web. De esta forma se logra una mayor eficiencia en tiempo y recursos para que el vendedor reemplace su manera tradicional de visualizar la información en tablas en Excel y en el sistema MBA3.

Para la predicción en series de tiempo, los datos tienen que ser estacionarios. En el análisis exploratorio de datos se puede ver que las ventas siempre fueron variables a medida que pasaba el tiempo, incluso llegando a tener picos bajos en el año 2020. Por tal razón se tuvieron que transformar estos datos en estacionarios, para que la media variable se mantenga constante en el tiempo. Este es uno de los requisitos básicos para usar series de tiempo. Uno de los métodos más fiables para determinar estacionalidad es el test de Dickey-Fuller, que establece una hipótesis nula que indica que una serie es no estacionaria. La hipótesis puede ser rechazada cuando el valor de test estadístico es menor que los valores críticos; y cuando el valor -p, es menor a 0.05. Una vez que se aplicaron métodos de transformación en los datos, se usó nuevamente el test de Dickey-Fuller demostrando que esta vez los datos sí son estacionarios. Estos datos estacionarios fueron usados en el modelo de predicción.

En el proyecto integrador se realizó un procesamiento importante con los datos que actualmente la empresa tiene, por lo que analizando esos datos se pudo concluir que siempre será mejor la calidad de los datos que la cantidad. Esto se debe a que, se puede tener muchísima información, pero esta puede ser redundante o sin valor, lo que provocaría que en un futuro un modelo de predicción no funcione correctamente. Esto puede ser controlado desde el punto de minería de datos, que usa la información cruda, la procesa, la limpia y la transforma; para que así todos estos datos sean consistentes y tengan un valor importante en su análisis. Por tal razón, se puede decir que para problemas que requieran el uso de inteligencia artificial, ya sea de tipo de aprendizaje supervisado o no, es importante saber que la calidad de los datos siempre será determinante en la creación o evaluación del modelo. En el proyecto integrador los datos iniciales se obtuvieron desde un ERP; estos datos fueron tratados, limpiados y procesados para que sean útiles al momento de querer usarlos en una red neuronal. Los resultados de este proceso de minería de datos fueron los esperados ya que, con estos mismos datos, se pudo usar y entrenar una red neuronal, capaz de predecir las ventas futuras de la compañía con una alta precisión.

Las redes neuronales son una herramienta potente en el mundo del aprendizaje supervisado, con la ayuda de librerías como Keras, la construcción de un modelo de red neuronal se puede simplificar bastante. En el proyecto, se usó una red neuronal LSTM, con dos capas dense para poder predecir las ventas de la compañía. Las capas dense son capas fully-connected, donde cada neurona tiene una conexión con la capa anterior, a las que se le tiene que especificar cuántas neuronas tiene y qué tipo de función de activación va a ejecutar. Para la capa de entrada se definió una capa LSTM con 4 neuronas; en la capa oculta, se definieron 2 neuronas con una función de activación *tanh;* y en la capa de salida solamente se añadió una neurona, sin función de activación (ya que las capas de salida no necesitan). El optimizador que se configuró en la red neuronal fue el descenso de gradiente estocástico, que es un

optimizador muy eficaz en algoritmos de aprendizaje supervisado. Los resultados que se obtuvieron fueron bastante buenos ya que en las pruebas de R2, el coeficiente alcanzado fue de 0.93; siendo 1 el mejor valor posible; para la métrica MAPE en cambio se obtuvo un puntaje de 0.024, lo cual es bastante aceptable ya que esta métrica indica que, cuanto menor sea este coeficiente, mejor será el modelo para predecir resultados. Todo esto fue posible primero haciendo una correcta limpieza en los datos desde el inicio, luego procesando estos datos, y luego pasando los datos a una red neuronal configurada.

REFERENCIAS BIBLIOGRÁFICAS

Aigiomawu, E. (Octubre 2018). Analyzing time series data in Pandas. Recuperado el 23 de Abril de 2023 desde https://towardsdatascience.com/analyzing-time-series-data-in-pandas-be3887fdd621

Brownlee, J. (Mayo 2017). How to Convert a Time Series to a Supervised Learning Problem in Python. Recuperado el 25 de abril desde https://machinelearningmastery.com/convert-time-series-supervised-learning-problem-python/

Decide. (Agosto 2019). “Calidad o cantidad de datos, qué es más importante para la IA”. Recuperado el 27 de enero de 2023 desde https://decidesoluciones.es/calidad-o-cantidad-de-datos-para-ia/

Del Tena, R. (Septiembre 2020). ¿Qué es el SKU y para qué sirve? Recuperado el 20 de Octubre de 2022 desde https://www.holded.com/es/blog/que-es-el-sku

Easysoft. (s/f). “Connecting to ODBC Databases from Python with pyodbc”. https://www.easysoft.com/developer/languages/python/pyodbc.html

Jain, A. (Febrero 2016). A Comprehensive Beginner’s Guide to Creating a Time Series Forecast. Recuperado el 28 de Abril de 2023 desde https://www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/?

Muñoz, A. y Salazar, F. (Julio 2022). Aplicación de redes neuronales LSTM al problema de predicción de la dirección de movimiento diario para bitcoin. Recuperado el 20 de enero de 2023 desde

https://www.academia.edu/49454684/%20APLICACION_DE_REDES_NEURONALES_LSTM_A_BITCOIN-libre.pdf

Martínez, J. (Enero 2020**).** ¿Qué es un Optimizador y Para Qué Se Usa en Deep Learning? *Datasmarts.* Recuperado el 20 de Mayo de 2023 desde https://datasmarts.net/es/que-es-un-optimizador-y-para-que-se-usa-en-deep-learning/#:~:text=Entonces%2C%20lo%20que%20un%20optimizador,se%20conoce%20como%20%E2%80%9Cbackpropagation%E2%80%9D.

Matiach, D. (Marzo 2001). "Redes neuronales: conceptos básicos y aplicaciones". Recuperado el 20 de enero de 2023 desde redesneuronales-libre.pdf (d1wqtxts1xzle7.cloudfront.net)

Méndez, E. (Mayo 2023). Ingeniería de datos para pronóstico de ventas e integración web/DB de los datos completos de productos en venta. Recuperado el 22 de Mayo desde https://github.com/EdisonMendez/proyectoIntegradorCmp

Sandoval, P & Vallejos, J. (diciembre 2022). Análisis y modelamiento de los márgenes por negocios de la empresa derco mediante series temporales y redes neuronales recurrentes lstm. Recuperado el 2 de Febrero de 2023 desde https://repositorio.udd.cl/server/api/core/bitstreams/4617d7e6-1dbc-461c-ac2e-0e4fed4fcaa0/content