UNIVERSIDAD DE ANTIOQUIA

# Predicción De Las Materias Primas Que Deben Presupuestarse De Acuerdo A Las Ventas De Una Compañía Farmacéutica


Autora:

Zuleima Lopera Arango


Tesis o trabajo de investigación presentada(o) como requisito parcial para optar al título de:

**Especialista en Analítica y Ciencia de Datos**


Asesor:

Jhon Jair Quiza Montealegre

Universidad de Antioquia

Facultad de Ingeniería – Departamento de Ingeniería de Sistema

Medellín, Colombia


Año 2022

| | |
|---|---|
| **Cita** | (Lopera Arango, 2022) |
| **Referencia**<br><br>**Estilo APA 7 (2020)** | Lopera Arango. (2022). *Predicción De Las Materias Primas Que Deben Presupuestarse De Acuerdo A Las Ventas De Una Compañía Farmacéutica,* [Trabajo de grado especialización]. Universidad de Antioquia, Medellín, Colombia. |



Especializa en Analítica y Ciencia de los Datos, Cohorte III

UNIVERSIDAD DE ANTIOQUIA
Vicerrectoría de Docencia

Sistema de Bibliotecas

Centro de Documentación Ingeniería CENDOI

**Repositorio Institucional:** http://bibliotecadigital.udea.edu.co

Universidad de Antioquia - www.udea.edu.co

Rector: John Jairo Arboleda Céspedes

Decano: Jesús Francisco Vargas Bonilla

Jefe De Departamento: Diego José Luis Botia Valderrama

**Tabla de contenido**

## 1. Resumen

El objetivo del proyecto es realizar un modelo de predicción de presupuesto de materias primas de acuerdo a las ventas en años anteriores de un producto específico.

Por ejemplo, si se tiene en cuenta las cantidades facturadas del producto, el modelo debe ser capaz de predecir las cantidades que se venderán al año siguiente y de esta manera tener un presupuesto de las materias primas para esas cantidades predichas.

El proyecto se desarrolla con el fin de generar mejores presupuestos en cuanto a las materias primas que deben ser solicitadas para la fabricación de un producto de acuerdo a sus ventas

En estos momentos, en la compañía se tienen predicciones manuales acerca de las cantidades que se venderán y se generan presupuestos y órdenes de compra de materias primas que en la mayoría de ocasiones no son correctas y de esta manera se ven afectados los despachos ya que se presentan faltantes de materias prima, o compras en excesos que ocupan más espacio en bodega.

La idea es generar un modelo que realice la predicción de ventas para presupuestar materias primas, donde inicialmente se realice la limpieza de los datos, para evitar errores en los algoritmos, luego dividir en entrenamiento y validación, e incluir modelos de series de tiempo que realicen predicciones mensuales de la venta de la empresa y finalmente verificar los resultados de cada modelo y de acuerdo a las métricas, elegir el que mejores resultados obtenga.

Cuando se tienen series de tiempo, es elemental invertir tiempo en analizar la variable que se desea pronosticar. Se deben entrar a considerar: la estacionariedad, la estacionalidad, las distribuciones y las relaciones de características externas para el diseño de la arquitectura de cualquier modelo.

Los modelos de series de tiempo estructurales incluyen ideas estándares del modelado como procesos autorregresivos, medias móviles, tendencias lineales locales, estacionalidad, regresiones y selección de variables y covariables externas, en donde se incluyen series con una relación alta con la serie de interés.

Para el presente proyecto se realizaron modelos para predecir las ventas de seis productos de una compañía farmacéutica y tener la disponibilidad de materias primas en la demanda. Los mejores resultados que arrojaron el menor error en la validación, se obtuvieron con modelos de Deep Learning conformados con redes neuronales en capas Dense usando la herramienta de *TensorFlow* a través de la librería de *keras*. Las ventajas en la empresa de usar este tipo de predicciones es que se disminuye las ventas perdidas, mejora el servicio al cliente por la disponibilidad de los productos y disminuye inventarios obsoletos.

**Repositorio:**

https://github.com/Zuleimaloperaa/MONOGRAFIA.git

## 2. Descripción del Problema

### 2.1 Problema de Negocio

Las predicciones manuales de ventas en la compañía que pertenece a la industria farmacéutica, no son certeras y afectan el despacho de los productos hacia los clientes, ya que, en ocasiones, no se tienen las materias primas suficientes para abastecer al equipo de ventas, o por el contrario, la compra en exceso, ocupa el almacenamiento en bodega de gran volumen de materias que no se usan en el tiempo suficiente a su vencimiento, ocasionando pérdidas. Por esta razón se desea implementar un algoritmo de Machine Learning que, de acuerdo a las ventas en años

anteriores, sea capaz de predecir las ventas futuras y de esta manera tener un presupuesto de materias primas para poder ser solicitadas a tiempo y cumplir con la demanda en las ventas.

## 2.2 Aproximación desde la Analítica de Datos

A pesar de que los eventos futuros presentan incertidumbre, el pronóstico es parte fundamental de la planificación del futuro. De esta manera, las empresas necesitan pronosticar las demandas de consumo para mantener un inventario suficiente de sus productos.

Un modelo de series de tiempo se rige por una suposición estructural particular. Por ejemplo, un componente podría codificar un efecto estacional, otro la tendencia lineal local, otro una dependencia lineal local y otro una dependencia lineal de otro conjunto de series temporales.

Al realizar algoritmos de modelos realizando suposiciones sobre los procesos que generan los datos, las series de tiempos pueden producir, en su mayoría, pronósticos razonables. Estos modelos ayudan a interpretar fácilmente las predicciones de los datos pasados y pronósticos futuros en componentes estructurales, además de utilizar una formulación probabilística que puede manejar los datos faltantes y proporcionar cuantificación de la incertidumbre basado en principios.

En series de tiempo, se usan modelos dinámicos para las estimaciones por medio de datos pasados con componentes autoregresivas, integradas y promedios móviles como los modelos de ARIMA. Por otro lado, existen modelos de Machine Learning como Vectores de Soporte de Regresión, minimizando el error e individualizando el hiperplano que maximiza el margen, teniendo en cuenta que se tolera parte del error. Sin embargo, en pronósticos de ventas, las Redes Neuronales Artificiales (RNA) presentan una mayor precisión en comparación con los métodos de regresión lineal y ARIMA (Morales Castro et al., 2019). Estas redes neuronales se implementan con herramientas informáticas como *TensorFlow* a través de la librería de *keras*, creando modelos de aprendizaje profundo que durante el entrenamiento permite interactuar con la arquitectura de la

red, cambiando la cantidad de capas y de neuronas, así como el tamaño del batch y las épocas para lograr mejores predicciones (Camacho, 2020). Existen varios tipos de capas, desde las más sencillas como las Dense son las capas que conectan cada neurona en una capa con todas las salidas de la capa anterior, o capas LSTM que es un tipo de red neuronal recurrente que presentan grandes arquitecturas para el entrenamiento exitoso.

### 2.3 Origen de los Datos

Los datos usados en el análisis son proporcionados por una compañía farmacéutica. Contienen:

- DISTRITO:
- ZONA
- FACTURA
- TIPOCLIENTE
- IDCLIENTE
- CANTFACTURADA:
- PIEZA
- FECHAFACTURA:
- MESFFACTURA
- ANOFACTURA
- NROORDEN
- FECHAORDEN
- TIPOPIEZA

Las columnas de interés filtradas para el proyecto son: la cantidad facturada, la pieza (código interno de la presentación del producto) y la fecha facturada. El producto no se revela por motivos de confidencialidad. Estos datos corresponden a las ventas entre el 2015 y el 2021.

### 2.4 Métricas de Desempeño

Las métricas del proyecto son las siguientes:

- La métrica general para evaluar el desempeño del proyecto, se realizó comparando el error del modelo generado en el actual proyecto y el error del pronóstico que se usa en la compañía. De esta manera se establecerán las métricas del negocio.
- La métrica de Machine Learning que se usa para evaluar el desempeño de los modelos es el porcentaje medio del error absoluto MAPE, que es una relación media entre el error absoluto y el valor absoluto y permite dar una idea del tamaño de los errores en comparación con los valores. El MAPE, mide la precisión como un porcentaje. Esta es la medida más común para pronosticar el error ya que las unidades de la variable se escalan a unidades porcentuales y facilita la comprensión.

La ecuación que representa al MAPE es la siguiente:

$$MAPE = \frac{100}{N} \times \sum_{i=1}^{N} \left| \frac{x_i - \hat{x}_i}{x_i} \right|$$

Donde:

- $x_i$ son las observaciones actuales de las series de tiempo
- $\hat{x}_i$ son las series de tiempo estimadas o pronosticadas.
- $N$ es el número de punto de datos no faltantes.

El menor MAPE del pronóstico realizado por la compañía, versus el MAPE pronosticado por el modelo TensorFlow, es el que se usará para la planeación de la demanda.

# 3. Datos

## 3.1 Datos Originales

Como se mencionó anteriormente, el dataset está conformado por 13 columnas, pero las columnas de interés realmente son tres, que son:

- FECHAFACTURA: El formato de esta columna es aaaa-mm-dd y es la que constituye el orden de la serie de tiempo, por esta razón durante la primera etapa de la limpieza de datos se estipula como el índice de las filas.
- PIEZA: Cada producto de la compañía, de acuerdo a su presentación, se bautiza con un número de pieza, y cada pieza constituye un único modelo por su comportamiento en las ventas.
- CANTFACTURADA: Las cantidades facturadas de cada pieza es el número de piezas vendidas en cada instante de tiempo

El dataset cuenta con un total de 701262 registros, después de eliminar un 0.447% de datos eliminados por estar nulos y no implicar una pérdida significativa en los datos.

El formato en el que se encuentran los datos es un archivo en Excel ubicado en una ruta de Drive que tiene acceso libre para cualquier persona con el enlace. A través del código de Python, se tiene el enlace que automáticamente accede al archivo.

## 3.2 Datasets

Para la preparación de los datos para ingresar al modelo, se tuvo que seleccionar inicialmente las tres columnas de interés que comprendían la fecha de factura, el nombre de la

pieza y la cantidad facturada. Se instanció la fecha como índice y con formato de fecha reconocido por Python. Por otro lado, como la pieza se conforma por letras y números, para evitar errores de digitación entre mayúsculas y minúsculas, se colocan todos los datos en mayúscula.

Cada pieza tiene entre 1, 2 o 3 productos iguales, que difieren en una letra al final de ellas. Esta letra hace referencia al tipo de comercialización. Sin embargo, deben tratarse con modelos diferentes porque cada uno de ellos implica materias primas diferentes en cuanto al empaque y los costos.

Pero para definir la eficiencia del modelo de Deep Learning generado para el este proyecto, versus la forma actual de pronosticar en la compañía, se deben agrupar al final cada modelo generado por las piezas con el mismo nombre y se deben sumar ya que la compañía genera predicciones para las piezas sin tener en cuenta el tipo de comercialización, es decir, sin tener en cuenta la última letra de la Pieza.

**Figura 1**

Explicación de las métricas.


La Figura 2 presenta las piezas más vendidas, con la cantidad de piezas iguales que difieren en el tipo de comercialización.

**Figura 2**

Piezas iguales comercializadas.

| | PIEZA | PIEZASIGUALES | CANTIGUALES |
|---|---|---|---|
| 0 | PT02025R | [PT02025R, PT02025I] | 2 |
| 1 | PT02031R | [PT02031R, PT02031I] | 2 |
| 2 | PT02025I | [PT02025R, PT02025I] | 2 |
| 3 | PT01182R | [PT01182A, PT01182R] | 2 |
| 4 | PT02022R | [PT02022R, PT02022I] | 2 |
| 5 | PT02032R | [PT02032R, PT02032I] | 2 |
| 6 | PT58020A | [PT58020A, PT58020I, PT58020B] | 3 |
| 7 | PT33169A | [PT33169A] | 1 |
| 8 | PT00528M | [PT00528M] | 1 |
| 9 | PT01008A | [PT01008A, PT01008M, PT01008I] | 3 |

Se generó una función que agrupa los datos por meses, ya que los pronósticos de la empresa se dan de esta manera.

La partición de los datos en entrenamiento y validación se realizó a partir de una función de partición, que dividía los datos en 80% entrenamiento y 20% validación. Como se pronosticaron 6 piezas de la empresa, cada uno de ellas tenía tamaños diferentes en entrenamiento y validación.

**Figura 3**

Cantidad de datos de Entrenamiento y Validación para cada pieza.

| Productos | Cantidad_Entrenamiento | Cantidad_Validacion |
|---|---|---|
| PT01025A | 67 | 17 |
| PT02025R | 56 | 15 |
| PT02025I | 56 | 14 |
| PT59202I | 36 | 10 |
| PT59202A | 66 | 17 |
| PT01002A | 67 | 17 |

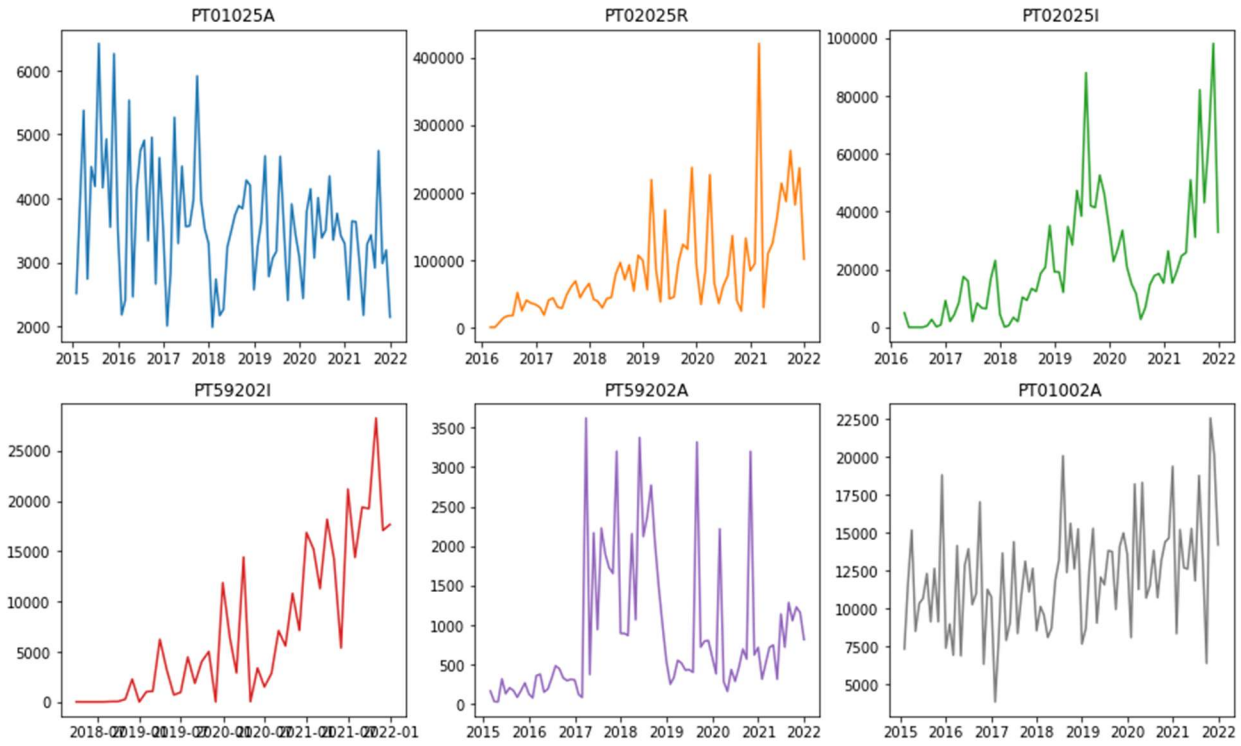
### 3.3 Descriptiva

Se realizó el modelo de seis Piezas de la compañía, teniendo encuenta que se encuentran entre las 15 más vendidas en el año 2021. Las piezas son : PT01025A, PT02025R, PT02025I, PT59202I, PT59202A, PT01002A.

A continuación, se presenta el gráfico de la cantidad facturada de cada pieza con respecto al tiempo:

**Figura 4**

Gráficos de cantidades facturadas versus tiempo de las Piezas




En la Figura anterior, se observa que no hay una tendencia visualmente estacionaria en el comportamiento de las ventas de las piezas y que esto dificulta las predicciones convencionales, por esta razón se usaron modelos de Deep Learning usando *TensorFlow* para predecir el comportamiento en las ventas futuras de la compañía. Sin embargo, para verificar la estacionalidad de la serie, se usaron herramientas de *statsmodel* como la prueba de Dickey- Fuller aumentada en donde, si el valor p es mayor a 0.05, no se rechaza la hipótesis nula y los datos no son estacionarios y si p es menor a 0.05 la hipótesis nula se rechaza y los datos son estacionarios. Al probar dicha función en los datos de cada pieza, se tiene que ninguna serie es estacionaria, excepto la pieza PT59202A.

Por otro lado, en la Figura 4 representa el resumen estadístico para las piezas, donde se observa el número de datos después de agrupar por meses, el promedio, la desviación estándar y

los cuartiles. Los valores altos en las desviaciones estándares, representan la dispersión de las cantidades mensuales facturadas a través de los meses.

**Figura 5**

Resumen estadístico de cada pieza.

| | PT01025A | PT02025R | PT02025I | PT59202I | PT59202A | PT01002A |
|---|---|---|---|---|---|---|
| **count** | 84.000000 | 71.000000 | 70.000000 | 46.000000 | 83.000000 | 84.000000 |
| **mean** | 3610.380952 | 85072.154930 | 21249.414286 | 7026.282609 | 892.409639 | 12098.309524 |
| **std** | 967.287773 | 74626.631573 | 21112.058409 | 7490.447572 | 893.921370 | 3606.597549 |
| **min** | 1986.000000 | 574.000000 | 0.000000 | 0.000000 | 25.000000 | 3862.000000 |
| **25%** | 2982.750000 | 37400.500000 | 5352.250000 | 761.500000 | 307.000000 | 9134.000000 |
| **50%** | 3494.000000 | 60800.000000 | 16390.000000 | 4234.000000 | 540.000000 | 11996.000000 |
| **75%** | 4154.000000 | 104280.500000 | 30421.000000 | 13661.750000 | 1102.000000 | 14119.000000 |
| **max** | 6422.000000 | 420679.000000 | 98048.000000 | 28248.000000 | 3621.000000 | 22578.000000 |

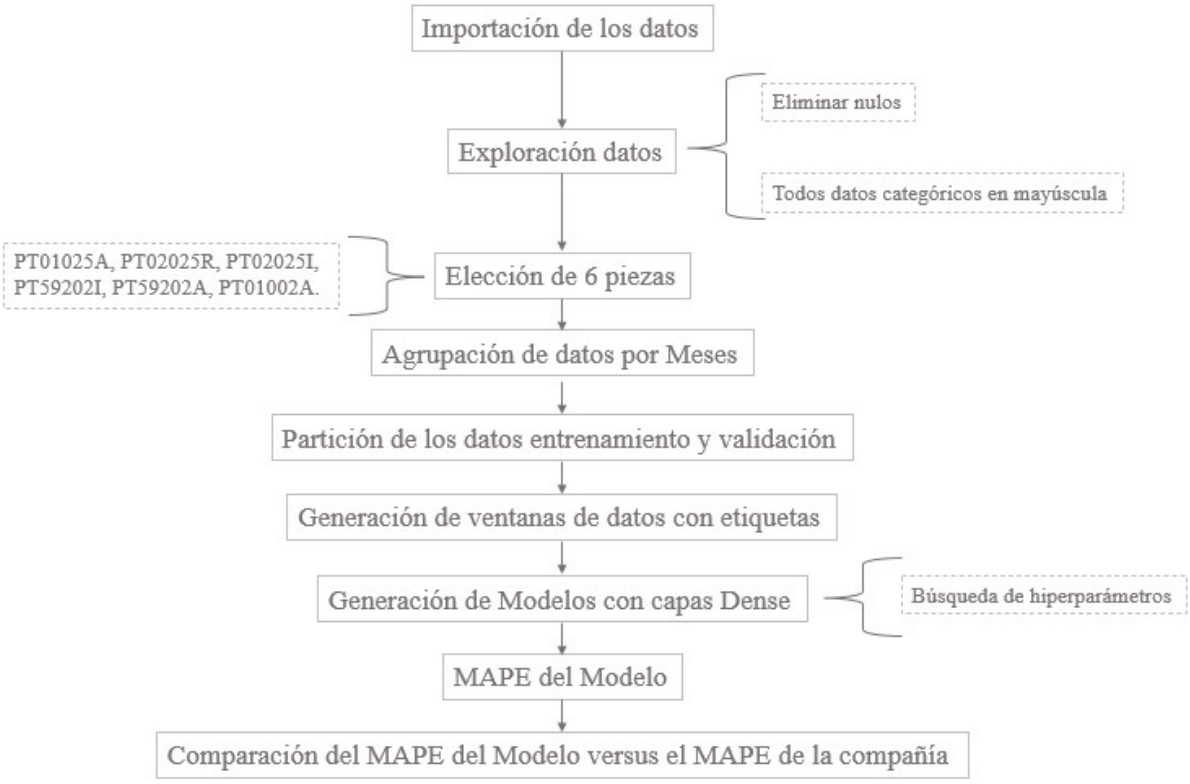
## 4. Proceso de Analítica

### 4.1 Pipeline Principal

El siguiente diagrama representa los pasos que se siguieron para la generación de los modelos de cada Pieza:

**Figura 6**

Pipeline principal




En el numeral 3.2 DATASETS se realiza una explicación del flujo de trabajo general con los datos.

## 4.2 Preprocesamiento

Para el preprocesamiento de los datos era necesario agrupar los datos en meses, por dos razones, la primera porque en un solo día se podía facturar varias veces la misma pieza o podía no haberse facturado; y la segunda, para poder ser comparado con la manera actual de predecir los datos de ventas en la compañía, ya que se hace de manera mensual.

Luego de partir los datos en entrenamiento y validación, se preparan para ingresar al modelo, con una función que en la que ingresan la serie, junto con los parámetros para el tamaño

de ventana y el lote que se usará para entrenar y el buffer aleatorio que determina como se van a barajar los datos. De esta manera, se dividen los datos de entrenamiento en características y etiquetas, en donde las características corresponden al tamaño de la ventana de una serie y las etiquetas es el valor siguiente. El tamaño final de los datos de entrenamiento y validación se pueden observar en la Figura 3, del numeral 3.2.

### 4.3 Modelos

Se experimentó en los datos de entrenamiento de cada pieza, para encontrar la arquitectura adecuada para cada modelo y sus hiperparámetros, hasta obtener el rendimiento más adecuado, medido usando el conjunto de datos de la validación. A continuación, se muestran las Figuras de las arquitecturas de los modelos de Tensor Flow con capas Dense, que mejor resultados generaron:

**Figura 7**

Arquitectura del Modelo para la Pieza PT01025A

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 6)                 78

 dense_4 (Dense)             (None, 6)                 42

 dense_5 (Dense)             (None, 1)                 7

=================================================================
Total params: 127
Trainable params: 127
Non-trainable params: 0
_________________________________________________________________
```

**Figura 8**

Arquitectura del Modelo para la Pieza PT02025R

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 64)                832

 dense_1 (Dense)             (None, 6)                 390

 dense_2 (Dense)             (None, 1)                 7

=================================================================
Total params: 1,229
Trainable params: 1,229
Non-trainable params: 0
_________________________________________________________________
```

**Figura 9**

Arquitectura del Modelo para la Pieza PT02025I

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_4 (Dense)             (None, 64)                832

 dense_5 (Dense)             (None, 64)                4160

 dense_6 (Dense)             (None, 8)                 520

 dense_7 (Dense)             (None, 1)                 9

=================================================================
Total params: 5,521
Trainable params: 5,521
Non-trainable params: 0
_________________________________________________________________
```

**Figura 10**

Arquitectura del Modelo para la Pieza PT59202I

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_4 (Dense)             (None, 16)                208

 dense_5 (Dense)             (None, 8)                 136

 dense_6 (Dense)             (None, 1)                 9

=================================================================
Total params: 353
Trainable params: 353
Non-trainable params: 0
_________________________________________________________________
```

**Figura 11**

Arquitectura del Modelo para la Pieza PT59202A

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 32)                416

 dense_4 (Dense)             (None, 8)                 264

 dense_5 (Dense)             (None, 1)                 9

=================================================================
Total params: 689
Trainable params: 689
Non-trainable params: 0
_________________________________________________________________
```

**Figura 12**

Arquitectura del Modelo para la Pieza PT01002A

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_6 (Dense)             (None, 30)                390

 dense_7 (Dense)             (None, 8)                 248

 dense_8 (Dense)             (None, 1)                 9

=================================================================
Total params: 647
Trainable params: 647
Non-trainable params: 0
_________________________________________________________________
```

La característica principal de estos modelos es que están constituidos con capas Dense, y que no se usan capas RNN, LSTM ni convoluciones. Aunque se realizaron iteraciones en modelos con arquitecturas que incluía este tipo de capas, la escasa longitud de las series de tiempo del proyecto, arrojaban errores altos.

En algunos de los modelos el ajuste del hiperparámetro de la tasa de aprendizaje, se realizó callbacks, que realizaba devolución de llamadas al final de cada época y era posible trazar un gráfico de las épocas contra la tasa de aprendizaje y elegir el punto más bajo de la curva como la tasa de aprendizaje ideal.

Se realizaron cambios en el tamaño de la ventana y los tamaños de lote para evaluar el MAPE de los modelos.

**Figura 13**

Resultados en el MAPE, en cambios del tamaño de ventana y de lote.

```
window_size = 6
batch_size = 10
shuffle_buffer_size = 67
```

| | Productos | MAPE_COMPANIA | MAPE_MODELO_TENSORFLOW |
|---|---|---|---|
| 0 | PT01025 | 71.818670 | 18.318977 |
| 1 | PT02025 | 75.297707 | 62.909943 |
| 2 | PT59202 | 60.001535 | 78.343582 |
| 3 | PT01002 | 53.787569 | 30.840067 |

```
window_size = 24
batch_size = 15
shuffle_buffer_size = 67
```

| | Productos | MAPE_COMPANIA | MAPE_MODELO_TENSORFLOW |
|---|---|---|---|
| 0 | PT01025 | 71.818670 | 16.859047 |
| 1 | PT02025 | 75.297707 | 65.082458 |
| 2 | PT59202 | 60.001535 | 36.442863 |
| 3 | PT01002 | 53.787569 | 35.907070 |

```
window_size = 18
batch_size = 6
shuffle_buffer_size = 67
```

| | Productos | MAPE_COMPANIA | MAPE_MODELO_TENSORFLOW |
|---|---|---|---|
| 0 | PT01025 | 71.818670 | 21.576509 |
| 1 | PT02025 | 75.297707 | 92.818459 |
| 2 | PT59202 | 60.001535 | 40.708916 |
| 3 | PT01002 | 53.787569 | 38.514133 |

```
window_size = 12
batch_size = 12
shuffle_buffer_size = 67
```

| | Productos | MAPE_COMPANIA | MAPE_MODELO_TENSORFLOW |
|---|---|---|---|
| 0 | PT01025 | 71.818670 | 19.459902 |
| 1 | PT02025 | 75.297707 | 51.245182 |
| 2 | PT59202 | 60.001535 | 48.657066 |
| 3 | PT01002 | 53.787569 | 35.744728 |

De acuerdo a la figura anterior, se observa que para el producto PT01025 el tamaño ideal de ventana es de 24 y el lote de 15, mientras que el producto PT02025 el tamaño de ventana es 12 al igual que el lote, el producto PT59202 el tamaño de ventana ideal es de 24 y el lote de 15 y finalmente para el producto PT01002 el tamaño de la ventana es 6 mientras que el lote es 10.

### 4.4 Métricas

Para el cálculo de las métricas, se usaron las métricas de *keras* de *TensorFlow* con el Error Porcentual Absoluto Medio (MAPE), que a través del código se ve de la siguiente manera: *tf.keras.metrics.mean_absolute_percentage_error(x_valid, x_predict_model).numpy().*

En las métricas de negocio se compararon los datos predichos por la compañía usando la misma librería de *keras* para el MAPE.

# 5. Metodología

## 5.1 Baseline

Para el Baseline se tuvo en cuenta, como se mencionó anteriormente, modelos de Tensor Flow con capas Dense y una cantidad reducida de neuronas, ya que, debido a la escasez de datos proporcionados, elaborar modelos complejos como RNN, LSTM y convoluciones, generaban errores altos en el MAPE.

## 5.2 Validación

Para el proceso de validación era necesario generar la predicción del 20% de los datos de cada serie de tiempo a través de la función forescat_funtion, que arrojaba el pronóstico y graficaba los resultados. Después de esto se generaba el MAPE del modelo para luego ser comparado por el MAPE de la compañía.

## 5.3 Iteraciones y Evolución

Se inició con arquitecturas que incluía RNN, por su flexibilidad y la capacidad de procesar todo tipo de secuencias, además de capas Lambda que permiten realizar operaciones arbitrarias para ampliar la funcionalidad del keras de *TensorFlow*. También se LSTM y convoluciones. Sin embargo, las capas Dense arrojaron los mejores resultados.

## 5.4 Herramienta

- Librerías de NumPy, Pandas y Matplotlib
- Librería de *keras* de *TensorFlow*
- Herramienta *adfuller* del módulo de *statsmodels.*

# 6. Resultados

## 6.1 Métricas

El MAPE de los modelos de TensorFlow en comparación con los de la compañía de acuerdo al tamaño de ventana y tamaño de lote con los menores errores, se observan a continuación:

**Figura 14**

Resultados del MAPE de la compañía y de los modelos de *TensorFlow* para cada PIEZA

| | Productos | MAPE_COMPANIA | MAPE_MODELO_TENSORFLOW |
|---|---|---|---|
| 0 | PT01025 | 71.818670 | 16.859047 |
| 1 | PT02025 | 75.297707 | 55.524189 |
| 2 | PT59202 | 60.001535 | 36.442863 |
| 3 | PT01002 | 53.787569 | 30.840067 |

*Nota: Los cambios en los valores del MAPE del modelo se deben al barajamiento de los datos cuando se organizan antes de ingresar al modelo, sin embargo, estos valores oscilan entre ma o menos 5% en el resultado

## 6.2 Evaluación Cualitativa

De acuerdo a los resultados del MAPE, para el modelo de *TensorFlow*, se observan resultados inferiores a los del MAPE de los pronósticos de la compañía, evidenciando que estos modelos son más eficientes que los métodos utilizados actualmente para comprar las materias primas necesarias para cada pieza. Es necesario aclarar que el modelo se irá alimentando de los datos facturados mensualmente y de esta manera optimizar el modelo.

### 6.3 Consideraciones de Producción

A continuación, se describen cada uno de los componentes para la puesta en producción de los modelos:

- Entramiento del modelo: Para el entrenamiento del model se usará una interfaz de línea de comandos (CLI). Esta pieza de software permitirá a un usuario entrenar modelos con diferentes parámetros y subirlos a un contenedor S3 en AWS después de serializarlos usando la librería joblib que permite guardar objetos únicos de python en archivos a través de la función dump.
- AWS S3: Es una herramienta de AWS que permita el almacenamiento de cualquier tipo de archivos y su posterior consumo a través de la API de AWS.
- AWS lambda: Esta herramienta permite ejecutar funciones en la nube usando diferentes disparadores (triggers).
- AWS CloudWatch: Esta herramienta nos permite el monitoreo de nuestro servicio en la nube.
- AWS API Gateway: Esta herramienta nos permite crear un punto de acceso con rutas para que los usuarios puedan consumir el servicio. Nótese que también hace las veces de disparador de la función lambda que se ejcutará al detectar una nueva petición HTTP en alguna de las rutas expuestas por la API Gateway.

Con esta infraestructura un usuario cualquiera podrá desde un cliente (llámese aplicación web o algún servicio embebido) hacer peticiones enviando una fecha y obteniendo la cantidad facturada. (Peña, 2022)

## 7. Conclusiones

- Para los modelos de series de tiempo con bajo volumen de datos, si se desea trabajar con librerías de *keras* de *TensorFlow*, es importante tener en cuenta que la red neuronal no debe ser muy compleja y evitar usar capas RNN, LSTM y convoluciones ya que con capas Dense se logran resultados con errores bajos y con menos gasto computacional.
- Comparando el MAPE de la compañía versus el MAPE del Modelo de *TensorFlow* generados para el presente proyecto, se tienen errores inferiores para las los pronósticos del *TensorFlow,* indicando que ayudarían a la compañía a aproximar las compras de materias primas y reducir faltantes o compras en exceso de las mismas.
- El agrupamiento de los datos por mes redujo considerablemente las muestras para el modelo. En proyectos posteriores, se puede evaluar el agrupamiento por semanas y exponer a la compañía los resultados que pueden generar un mejor desempeño.
- El mejoramiento en los pronósticos a través de estos modelos de redes neuronales, disminuye las ventas perdidas, mejora el servicio al cliente por la disponibilidad de los productos y disminuye inventarios obsoletos.

## 8. Referencias


Camacho, J. A. (2020, mayo 25). Redes Neuronales con Tensorflow y Keras. JacobSoft; Jacob Avila Camacho. https://www.jacobsoft.com.mx/es_mx/redes-neuronales-con-tensorflow-y-keras/

Keras | TensorFlow Core. (n.d). TensorFlow. Recuperado 2022, de https://www.tensorflow.org/guide/keras?hl=es-419

Morales Castro, A., Ramirez Reyes, E., & Rodríguez Albor, G. (2019). Pronóstico de ventas de las empresas del sector alimentos: una aplicación de redes neuronales. Semestre economico, 22(52), 161–177. https://doi.org/10.22395/seec.v22n52a7

Peña Juan Camilo (2022), Despliegue de un modelo de clasificación de tumores de cáncer de mama. Universidad de Antioquia.

Statsmodels.Tsa.Stattools.Adfuller — statsmodels. (s/f). Statsmodels.Org. Recuperado el 30 de mayo de 2022, de https://www.statsmodels.org/dev/generated/statsmodels.tsa.stattools.adfuller.html