

MAESTRÍA EN EXPLOTACIÓN DE DATOS Y GESTIÓN DEL CONOCIMIENTO

(MCD Virtual Cohorte 2022)

Laboratorio 3 - 2024

INFORME FINAL

Grupo: Deep Learning 3

**Integrantes** Caccianini, Antonela  
 Carranza, Carina  
 Estevez, Rodrigo

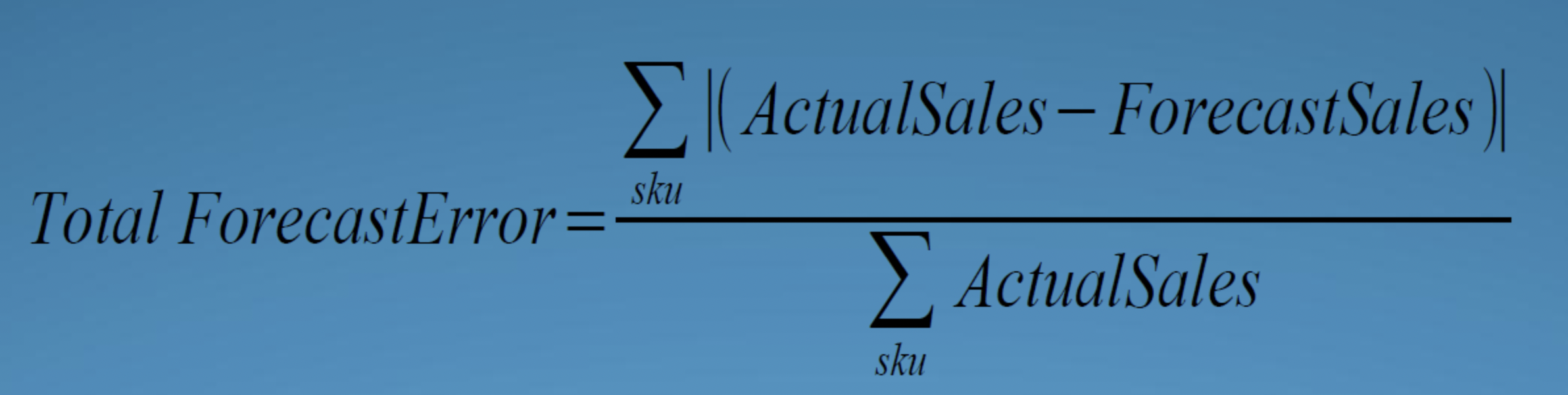
**Profesor**

Denicolay, Gustavo

1. **Business Problem**

El problema de negocio que se desarrolla en el presente informe consiste en el caso de una multinacional con una amplia gama de productos de consumo masivo desea implementar algoritmos de machine learning para optimizar sus pronósticos de demanda.

* El 02-enero se debe entregar el forecast de ventas para cada producto que se harán durante el mes 202002 , de forma que las plantas de la multinacional puedan fabricarlos durante el mes de 202001.
* La multinacional utiliza la siguiente métrica para evaluar el forecast recibido:



1. **Data Collection**

La información fue provista por la multinacional a través de las siguientes tablas:

* **Tabla productos\_a\_predecir.txt**: lista de productos (IDs) que deben predecirse para el mes 202002.
* **Tabla sell-in.txt.gz**: son las ventas sistematizadas de la siguiente manera:  
  -periodo: fecha de la venta   
  -customer\_id: identificación de cliente   
  -product\_id: identificación del producto   
  -plan\_precios\_cuidados: establecido por el gobierno nacional anunciándolo en el Boletín Oficial con una publicación para la fecha desde y otra posterior para la fecha hasta y nuestra empresa debe cumplirlo a rajatabla  
  -cust\_request\_qty: cantidad de solicitudes que el cliente hizo de ese producto, en ese mes.   
  -cust\_request\_  
  -tn: toneladas se vendió a ese cliente desde las 00:00:00 del primer dia del mes a las 23:59:59 del ultimo día de ese mes, la factura se emite y entrega con el camión en el portón de salida
* **Tabla tb\_stocks.txt**.gz: stock final de cada ítem al final de cada mes.
* **Tabla tb\_productos\_descripcion.txt**: tabla con productos y descripciones para identificar el mismo producto en distintos tamaños.

1. **Dataset review**

Al analizar los datasets provistos por la multinacional se realizaron las siguientes observaciones:

* + **Variables Cust\_request\_qty y Toneladas Pedidas:** cust\_request\_qty es la cantidad de solicitudes de un cliente para un producto en un mes y no representa la cantidad de toneladas.
  + **Discrepancias entre Stock Final y Ventas**: **A**lgunos registros muestran un stock final positivo, pero las ventas (tn) son menores que las solicitudes del cliente (cust\_request\_tn). Esto puede deberse a temas de logística y tiempos de entrega.Debido a lor previamente expuesto cust\_request\_tn y tn pueden diferir.
  + **Unidades de Medida en Stock y Ventas**: Todas las medidas están en toneladas, no en unidades.
  + **Período de Datos en Archivos de Ventas y Stock**: el archivo de ventas comienza en 2017 y el de stock en 2018 .La información histórica de stock está disponible solo desde 2018 debido a problemas de registro y backups.
  + **Códigos de Productos Diferentes para Productos Similares**: hay productos aparentemente idénticos (mismas categorías, marca y tamaño) tienen códigos de productos distintos. La diferencia se debe a un campo adicional (ejemplo donde etiquetas como "Ropas Finas" y "Ropas Oscuras" justifican diferentes códigos de productos).
  + **Famila de Productos:** La clasificación de productos se basa en relaciones de hermanos mayores y menores. no existe el concepto de "producto padre", sino "hermano mayor" y "hermano menor". (ej. el product\_id 20001 y el 20102 son el mismo producto en diferentes envases: uno en botella de plástico y otro en doypack para rellenar la botella).

1. **Data Pre-Processing considerations**

* **Predicción a nivel vs a nivel producto-cliente:** El requerimiento de la multinacional es predecir las ventas por producto correspondientes a la totalidad de los clientes. Para realizar dicha predicción, puede ser útil predecir por separado las ventas por producto por el cliente (al menos los 25 más grandes). Este approach se discutió utilizando como ejemplo el caso de la cadena Walmart. En el caso de nuestro grupo, no se optó por ese nivel de granularidad y se hicieron las predicciones a nivel producto. Mencionamos que no se pudo correr en la nube?
* **Imputación de ceros**
* **Ciclo de vida del producto**
* **Escalado por media y desvío**
* **Dividir el dataset por categoría de productos para su entrenamiento:**
* **Parametrizar weights:** el error total tiende a estar más influenciado por los errores en ventas grandes. Esto se explica porque las ventas grandes tienen un mayor impacto absoluto en el resultado del Total Forecast Error. En contrapartida, las ventas pequeñas, aunque pueden tener errores más altos, generalmente inciden menos en el total del error debido a que contribuyen menos en términos absolutos.Al utilizar weights permite enfocarse en los que más toneladas venden por mes..

1. **Model Building and Evaluation**

Los métodos con los cuales trabajamos a lo largo del proyecto

* **Promedio ventas de los últimos 12 meses:** estrategia ingenua cada producto en febrero va a vender el promedio del año anterior.
* **ARIMA /AutoARIMA (con/sin estacionalidad):** mostró peor resultado que la estrategia anterior de promedio de los últimos 12 meses.
* **Prophet / NeuralProphet:** este método también mostró una pobre performance.
* **LSTM**: La definición inicial fue optar por la especialidad de Deep Learning por lo cual decidimos realizar el análisis de la serie de tiempo utilizando un algoritmo LSTM (Long Short Term Memory).
* DeepAR: Modelo desarrollado por Amazon y diseñado para predecir series de tiempo utilizando RNN. Se utilizó el modelo DeepAR del paquete Gluonts, pero también hay otras opciones en otras bibliotecas.

1. **Code Analysis**

**A) Pre Procesamiento de los datos**

1.Se carga los datos de los archivossell-in.txt y prod\_a\_pred.txt. Se filtra de forma de conservar solo aquellos productos listados en prod\_a\_pred.txt.

2.Se agrupan las ventas por producto\_id y perido y suman las tn para cada item. El resultado es dataframe df\_sells con las toneladas totales de cada producto en cada periodo.

3. Se transforma el DF utilizando la función pivot para preparar los datos en un formato wide para ser utilizados en un modelo LSTM. Como resultado se obtiene un DF df\_lstm donde cada fila corresponde a un periodo, cada columna corresponde a un product\_id, y las celdas contienen los valores de tn correspondientes.

4. Se imputan los meses sin ventas de productos en 0 toneladas (solo en productos existeentes).

5. Se normalizan los datos en el DF df\_lstm utilizando StandardScaler utilizando la media y la desviación estándar calculados previamente. Se convierte el resultado de la transformación en un nuevo DF df\_lstm\_scale, manteniendo las mismas columnas y el mismo índice que el DF original df\_lstm.

6. Para completar con valores las ventas de productos que no existían previo a su lanzamiento se utilizará el promedio de ventas de los productos (valores escalados) en la categoría para cada período. Para esto se itera sobre combinaciones únicas de categorías (cat1 y cat2) en el DF descripción, calculan los valores promedio para los productos dentro de cada combinación, y utiliza estos promedios para rellenar valores faltantes (NaN) en el DF df\_lstm\_scale .

7. Se asigna una categoría a cada producto que representa la antigüedad del mismo. Para los productos nuevos que fueron lanzados dentro de los últimos 3 años se los imputa con su antigüedad real (cantidad de meses desde su lanzamiento) y con un valor negativo para los meses en donde no existía previamente. Luego se les asignó una categoría a los productos cuyas ventas decaen año a año (productos en proceso de discontinuación), y otra para los productos con ventas estables y más de 3 meses de antigüedad.

8. Las ventas del período 201908 se imputan por medio de interpolación debido a que en ese mes se descontinuaron las ventas de la multinacional.

9. Se calculan los promedios de las ventas absolutas para los periodos 201901, 201902, y 201903 en df\_lstm\_2 y los almacena en una lista llamada pesos.

10. Se utiliza la función create\_lagged\_features para crear un df de lagged features con lags de hasta los últimos 12 períodos. Con la función create\_mean\_features, se calculan las medias móviles de 2 a 5 periodos para la columna 'value' y las concatena en un nuevo DF.

**B) Creación de los conjuntos de entrenamiento, validación y prueba**

Modelo LSTM

11. Se crean los conjuntos de entrenamiento, validación y prueba (X\_train, X\_validate, X\_test, y\_train, y\_validate, y\_test) a partir de df\_modelo se verifica que las dimensiones de las matrices sean correctas y que no haya solapamiento o faltas en los datos.

Modelo DeepAR

El set de datos se introduce en el modelo y el modelo divide automáticamente los datos en train - test. Para introducir el dataset en DeepAR se debe convertir el índice en un formato datetime.

**C) Definición y Construcción del Modelo**

9. Se importa modelo LSTM:usando TensorFlow y Keras. Se ajustan de Parámetros para Reproducibilidad y se define el modelo:

* Establece una semilla para asegurar que los resultados del entrenamiento sean reproducibles.
* **N\_UNITS**: Número de unidades en cada capa LSTM en este caso 2000.
* **INPUT\_SHAPE**: Forma de la entrada del modelo, en este caso, 24 pasos de tiempo (horas) y 13 features.
* **Sequential()**: se crea un modelo secuencial.
* **LSTM(N\_UNITS, input\_shape=INPUT\_SHAPE, return\_sequences=True)**: se añade una capa LSTM con N\_UNITS unidades y la forma de entrada especificada. return\_sequences=True indica que la capa debe devolver la secuencia completa de salidas para la siguiente capa LSTM.
* **Dense(x\_tr.shape[2], activation='linear')**: se añade una capa densa con una activación lineal para la salida. El número de neuronas es igual al número de características de entrada, ya que queremos predecir valores continuos (regresión).
* **root\_mean\_squared\_error**: se define una función de pérdida personalizada que calcula la raíz del error cuadrático medio (RMSE).
* **RMSprop(learning\_rate=5e-4)**: se define el optimizador RMSprop con una tasa de aprendizaje de 5e-4. RMSprop adapta la tasa de aprendizaje para cada parámetro.
* **loss\_weights=pesos**: se especifica los pesos de la pérdida. pesos debe ser una lista o arreglo con los pesos correspondientes a las diferentes salidas del modelo (en caso de haber múltiples salidas). Si solo hay una salida, este parámetro puede omitirse.

D) **Entrenamiento del Modelo**

10. Se entrena el modelo durante 200 épocas, actualizando los pesos y ajustando el modelo para mejorar su precisión. Se utilizan los siguientes hiperparametros:

* **EPOCHS**: Se definió EPOCHS = 200. El modelo entrenará usando el conjunto de datos completo 200 veces.
* **BATCH\_SIZE**: Se definio BATCH\_SIZE = 5. El modelo actualizará sus parámetros después de procesar 5 ejemplos de entrenamiento.
* **modelo.fit()**: Este método entrena el modelo durante un número fijo de iteraciones (epochs) sobre los datos de entrenamiento.
* **validation\_data = (x\_vl, y\_vl)**: Datos de validación que se utilizarán para evaluar la pérdida y cualquier métrica del modelo al final de cada época.
* **verbose = 2**: Controla la verbosidad del entrenamiento. 2 significa que se imprimirá una línea por cada época.

**Proceso de Entrenamiento consiste en los siguientes pasos:**

* **Inicialización**:El modelo comienza con pesos aleatorios.
* **Forward Pass**: Para cada lote de tamaño BATCH\_SIZE, el modelo realiza un paso hacia adelante, calculando las predicciones en base a las entradas actuales.
* **Cálculo de la Pérdida**: La pérdida se calcula comparando las predicciones del modelo con las salidas reales.
* **Backward Pass (Retropropagación)**:El algoritmo de retropropagación calcula los gradientes de la pérdida con respecto a los pesos del modelo.
* **Actualización de Pesos**:Los pesos del modelo se actualizan utilizando el optimizador definido (RMSprop en este caso), que ajusta los pesos para minimizar la pérdida.
* **Repetición**:Este proceso se repite para todos los lotes de entrenamiento y durante todas las épocas especificadas.
* **Validación**:Después de cada época, el modelo se evalúa utilizando los datos de validación (x\_vl, y\_vl) para verificar cómo se está desempeñando el modelo en datos no vistos durante el entrenamiento.

**E)** **Evaluación del Modelo**

11. Se evalúa el rendimiento del modelo entrenado en los conjuntos de entrenamiento, validación y prueba utilizando la métrica de raíz del error cuadrático medio (RMSE) Se compara el desempeño del modelo en los tres conjuntos de datos (entrenamiento, validación y prueba).

* **RMSE Train**: bajo sugiere que el modelo está aprendiendo las relaciones en los datos de entrenamiento adecuadamente.
* **RMSE Val**: Un RMSE de validación significativamente mayor que el de entrenamiento puede indicar sobreajuste.
* **RMSE Test**: El RMSE del conjunto de prueba proporciona una medida final del rendimiento del modelo en datos completamente nuevos.
* Se puede determinar si el modelo está sobreajustado (alto RMSE de validación/prueba comparado con el de entrenamiento) o si tiene un buen desempeño general (RMSE de validación/prueba cercano al de entrenamiento).

**F) Predicción del modelo**

12. Se generan predicciones con el modelo LSTM y luego se realiza una transformación inversa de las predicciones para obtener los valores en la escala original. Los resultados se guardan en un DF con los product\_id y las predicciones transformadas a la escala original. Luego se exportan a un archivo CSV, excluyendo el índice del DF.

**Conclusiones**

Para el abordaje de la solución seleccionamos Deep Learning, para ello aplicamos LSTM y como subespecialidad deepAR.

Mediante los resultados obtenidos en las distintas pruebas evidenciamos la importancia del pre-procesamiento de datos, antes descrito y la aplicación del feature engineer. En este sentido aplicamos, lags del 1 a 12, suponiendo los lag inconclusos iguales a los del año siguiente. Se agregó un roll mean de los últimos 2 a 5 meses con el objetivo de evaluar si los clientes están consumiendo stock o acumulando stock. Además agregamos una variable que representa la antigüedad del producto vendido y si el mismo está en proceso de ser discontinuado.  
El error total arrojado en la competencia de Kaggle-Public-Leadboard fue de 0.273

En cuanto al modelo deepAR dió mejores resultados que LSTM significó menor tiempo de cómputo y la creación del modelo resultó de menor esfuerzo. La particularidad del modelo devuelve como resultado un sample de posibles valores futuros para cada serie, debiendo luego calcular el promedio para obtener la predicción. Esto permite calcular los intervalos de confianza de los valores predichos.   
El error total arrojado en la competencia de Kaggle-Public-Leadboard fue de 0.254

**Lecciones Aprendidas**

A continuación detallaremos aquellas pruebas que no arrojaron buenos resultados.

* Correr un LSTM, sin asignar pesos. Al escalar todas las series se le debe informar al modelo cuál es el peso que tiene cada serie en el error total. En el modelo deepAR utilizado no se encontró un parámetro para incorporar los pesos de las series.
* Error al imputar nulos, debido a que existe una dependencia temporal entre cada observación es importante no alterar el comportamiento de la serie al imputar los valores nulos.
* Feature consideradas para el dataset, se evidenció una mejora en la performance del modelo al agregar característica de la serie o de los productos.
* Modificar agosto 2019 ( época de elecciones) en NA, y luego por interpolación completarlos, resulta de suma importancia conocer el contexto de los datos. No hubiéramos advertido esta situación si no nos hubieran debidamente informado.
* Spacetime transformer, se intentó aplicar un transformer para realizar las predicciones pero nos resultó de suma dificultad poder avanzar en esta dirección. Las implementaciones que se encontraron de estos tipos de modelos fueron casos puntuales con casos muy diferentes a la problemática que tenemos.
* DeepAR, algoritmo de AMAZON al querer agregar más neuronas a las capas empieza a overfitear , siendo el seteo óptimo de parámetros importante encontramos que no necesariamente el modelo más complejo otorga mejores resultados. Además, para esta subespecialización existen muchas posibles distribuciones y fue difícil encontrar un paquete que no arroje conflictos con el entorno virtual.