



**UNIVERSIDAD
TORCUATO DI TELLA**

MASTER IN MANAGEMENT + ANALYTICS

**DEEP LEARNING PARA PREDICCIÓN DE
DEMANDA RETAIL A GRAN ESCALA: UN
MODELO INTELIGENTE POR PRODUCTO
PARA LA TOTALIDAD DE SUS TIENDAS**

TESIS

Santiago Gabriel Romano Oddone

Mayo 2025

Tutor: Federico Tomás Poncio

Resumen

La predicción de demanda en la industria del retail representa un desafío clave, especialmente en contextos de alta granularidad donde miles de combinaciones producto-tienda deben ser gestionadas simultáneamente. La precisión del pronóstico y la escalabilidad de los sistemas se vuelven factores críticos para garantizar una planificación eficiente del inventario y evitar quiebres de stock o excesos de mercadería.

Esta tesis presenta una metodología de modelado basada en deep learning, diseñada específicamente para abordar este escenario. La propuesta consiste en entrenar un único modelo por producto, utilizando de forma simultánea las series temporales de todas sus tiendas. Esta estrategia permite capturar patrones generales de comportamiento a nivel de producto, mejorar la calidad de las predicciones y reducir de forma considerable los tiempos de entrenamiento, evitando el enfoque tradicional de construir un modelo por cada combinación producto-tienda.

La solución fue validada mediante el desarrollo y evaluación de distintos modelos de deep learning bajo esta arquitectura, utilizando datos reales del entorno retail. Su desempeño se comparó frente a un sistema basado en promedios móviles y al algoritmo XGBoost, uno de los más utilizados en la industria. Los resultados posicionan al modelo Simple FeedForward como la alternativa más precisa y eficiente, consolidándose como una solución robusta y escalable para sistemas de predicción masiva en entornos productivos.

Índice

1. Introducción	6
1.1. Contexto	6
1.2 Problema	8
1.3 Objetivo	9
1.4 Estado del Arte	10
2. Datos	11
2.1 Introducción a los Datos	11
2.2 Análisis Exploratorio	11
2.3 Clusterización	15
2.4 Validación Problemática a Atacar	19
3. Metodología	20
3.1 Modelos de Predicción Propuestos y Benchmarks	20
3.1.1 Modelos Benchmark	21
3.1.2 Simple FeedForward: Multilayer Perceptron	21
3.1.2.1 Fundamento Teórico	22
3.1.2.2 Arquitectura y Funcionamiento	22
3.1.2.3 Implementación en GluonTS	22
3.1.2.4 Ventajas para el Retail	23
3.1.3 DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks	23
3.1.3.1 Fundamento Teórico	23
3.1.3.2 Arquitectura y Funcionamiento	24
3.1.3.3 Implementación en GluonTS	24
3.1.3.4 Ventajas para el Retail	25
3.1.4 Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting	25
3.1.4.1 Fundamento Teórico	26
3.1.4.2 Arquitectura y Funcionamiento	26
3.1.4.3 Implementación en GluonTS	26
3.1.4.4 Ventajas para el Retail	27
3.1.5 WaveNet: A Generative Model for Raw Audio	27
3.1.5.1 Fundamento Teórico	27
3.1.5.2 Arquitectura y Funcionamiento	28
3.1.5.3 Implementación en GluonTS	28
3.1.5.4 Ventajas para el Retail	29
3.2. Metodología de Entrenamiento y Validación	30
3.2.1 Definición de Conjuntos de Entrenamiento, Validación y Prueba	30
3.2.2 Entrenamiento por Producto	30
3.3. Metodología de Optimización de Modelos	31
3.3.1 Metodología de Selección de Hiperparámetros	31
3.3.1.1 Estrategia de Validación y selección	32
3.3.1.2 Random Search: Exploración Eficiente del Espacio de Parámetros	33
3.3.1.3 Optimización Bayesiana con TPE: Un Enfoque Adaptativo	33
3.3.2 Espacio de Hiperparámetros Considerado por Modelo	34

3.3.2.1 Simple FeedForward: Multilayer Perceptron	34
3.3.2.2 DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks	36
3.3.2.3 Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting	38
3.3.2.4 WaveNet: A Generative Model for Raw Audio	40
3.3.3 Control de Sobreajuste de Entrenamiento	42
3.4 Ingeniería de Atributos	43
3.5 Evaluación de Modelos	43
3.5.1 Métrica de Evaluación	43
3.5.2 Metodología de Evaluación	44
4. Resultados Obtenidos	48
4.1. Comparación Modelos vs Benchmark	48
4.1.1. Cluster 0	48
4.1.2. Cluster 1	52
4.1.3. Cluster 2	56
4.2. Comparación entre Todos los Modelos	60
4.3. Registro de Tiempo de Entrenamiento	61
4.4. Análisis de Resultados	62
4.5 Comparación Mejor Modelo vs XGBoost	63
4.6 Análisis de Hiperparámetros del Modelo Simple FeedForward	64
5. Conclusiones	66
6. Futuras Direcciones	67
Referencias	68

Índice de Figuras

Figura 1. Parámetros utilizados por Smart Replenishment	7
Figura 2. Etapas del proceso de reposición de mercadería mediante Smart Replenishment	8
Figura 3. Distribución de ventas según mes y dia de la semana	11
Figura 4. Distribución de tiendas y productos según su alcance comercial	12
Figura 5. 10 productos con mayor volumen de venta	13
Figura 6. Distribución de la venta diaria promedio por tienda	14
Figura 7. Distribución de la venta diaria promedio por producto	15
Figura 8. Método del codo para detectar número óptimo de clusters	16
Figura 9. Venta promedio del trimestre según cluster	16
Figura 10. Venta promedio del año según cluster	17
Figura 11. Distribución de productos según cluster	18
Figura 12. PCA análisis según dos y tres dimensiones	18
Figura 13. Análisis de demanda del producto y tienda particular	19
Figura 14. Representación del modelo de Multilayer Perceptron	21
Figura 15. Representación del modelo DeepAR	23
Figura 16. Representación del modelo Temporal Fusion Transformer	25
Figura 17. Representación del modelo WaveNet	27
Figura 18. Metodología de entrenamiento por producto	31
Figura 19. Desempeño de Simple FeedForward vs Benchmark - Cluster 0	48
Figura 20. Desempeño de DeepAR vs Benchmark - Cluster 0	49
Figura 21. Desempeño de Temporal Function Transformers vs Benchmark - Cluster 0	50
Figura 22. Desempeño de WaveNet vs Benchmark - Cluster 0	51
Figura 23. Desempeño de Simple FeedForward vs Benchmark - Cluster 1	52
Figura 24. Desempeño de DeepAR vs Benchmark - Cluster 1	53
Figura 25. Desempeño de Temporal Fusion Transformers vs Benchmark - Cluster 1	54
Figura 26. Desempeño de WaveNet vs Benchmark - Cluster 1	55
Figura 27. Desempeño de Simple FeedForward vs Benchmark - Cluster 2	56
Figura 28. Desempeño de DeepAR vs Benchmark - Cluster 2	57
Figura 29. Desempeño de Temporal Fusion Transformers vs Benchmark - Cluster 2	58
Figura 30. Desempeño de WaveNet vs Benchmark - Cluster 2	59
Figura 31. Relación entre RMSE mediano y tiempo de entrenamiento por cluster	61
Figura 32. Hiperparámetros utilizados por Simple FeedForward	64

Índice de Tablas

Tabla 1. Espacio de hiperparámetros considerado - Simple FeedForward	35
Tabla 2. Espacio de hiperparámetros considerado - DeepAR	37
Tabla 3. Espacio de hiperparámetros considerado - Temporal Fusion Transformers	40
Tabla 4. Espacio de hiperparámetros considerado - WaveNet	42
Tabla 5. Productos considerados para el análisis según cluster	46
Tabla 6. Análisis mediana RMSE de todos los modelos según cluster	60
Tabla 7. Análisis de ranking medio de todos los modelos según cluster	61
Tabla 8. Comparación del RMSE entre Simple FeedForward y XGBoost por cluster	63
Tabla 9. Comparación de la mediana del RMSE entre SFF y XGBoost por cluster	64

1. Introducción

1.1. Contexto

El sector retail en Brasil enfrenta desafíos cada vez más complejos en la gestión de inventarios y la optimización de la distribución de mercadería, especialmente para grandes fabricantes como CocaCola y Ambev, que operan en una gran cantidad de redes de supermercados a nivel nacional. Estas empresas deben garantizar que sus productos lleguen a los supermercados en las cantidades correctas para evitar tanto el desabastecimiento como el exceso de inventario, lo que implica gestionar una logística precisa y ajustada a las necesidades de cada tienda.

Scannotech, empresa líder en soluciones basadas en datos, ofrece Smart Replenishment, un sistema diseñado específicamente para grandes fabricantes que facilita su comunicación con las redes de supermercados en las que operan mediante la generación de sugerencias de envío de mercadería.

Funcionamiento de Smart Replenishment:

Para cada combinación de Tienda/Producto considerada, se configuran los siguientes parámetros en el sistema:

- Stock máximo y mínimo: Define los límites de la cantidad de inventario permitidos en la tienda.
- Pedido máximo y mínimo: Define los límites de la cantidad de pedido permitidos.
- Día de emisión y día de entrega: Define los días de la semana en que se emiten y entregan los pedidos.
- Días stock: Define la cantidad de días a cubrir de stock con el pedido.

Smart Replenishment tiene acceso a información de stock y de ventas de todos los supermercados en donde estos grandes fabricantes comercializan sus productos. Toda esta información es utilizada para el cálculo diario de las sugerencias de envío de mercadería desde la fábrica del cliente hacia las tiendas.

En la siguiente imagen se muestra cómo influyen estos los parámetros a la hora de calcular una sugerencia para cada combinación Tienda/Producto:

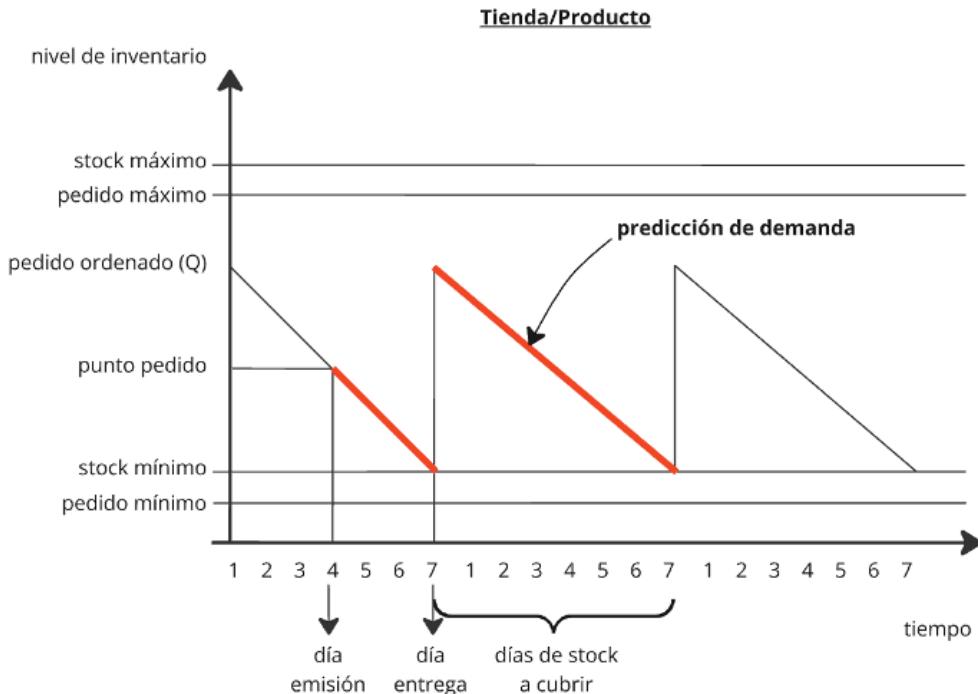


Figura 1. Parámetros utilizados por Smart Replenishment

Como se puede observar en la imagen, la combinación Tienda/Producto analizada tiene configurado un día emisión igual a 4, es decir Jueves, y un día entrega igual a 7, es decir Domingo. Los días de stock a cubrir con el pedido son 7 en este caso.

Los días Jueves, Smart Replenishment calcula la cantidad óptima a enviar del producto del fabricante a la tienda con el objetivo de poder cubrir los días de stock deseados, respetando las cotas de stock y pedidos máximo y mínimo.

Para esto, el sistema utiliza entre otras variables, **la predicción de la demanda del producto en la tienda desde el día en que se emite el pedido, hasta el último día de stock a cubrir**. En este ejemplo se utilizaría la predicción de demanda de 10 días.

De esta manera, el sistema genera recomendaciones que se envían a los fabricantes para que luego sean compartidas por ellos con las redes de supermercados en las que operan, quienes se encargan de aprobar los pedidos y ajustar las cantidades finales según corresponda. A continuación se muestra gráficamente las etapas del proceso:

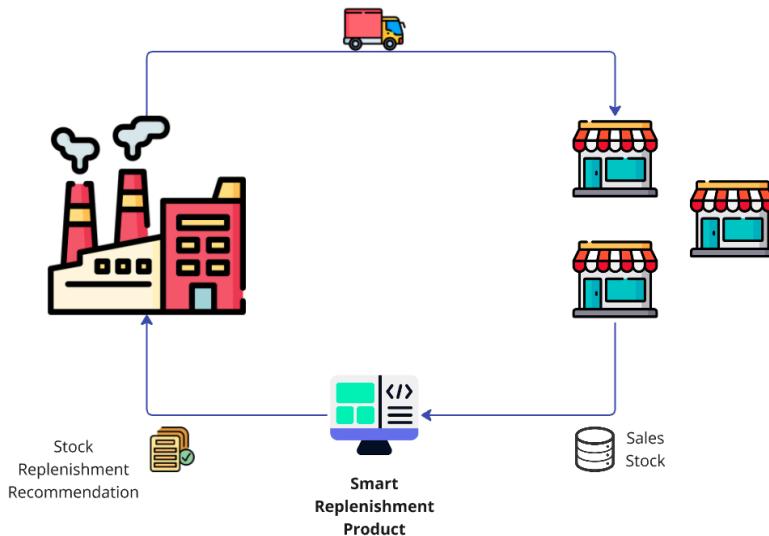


Figura 2. Etapas del proceso de reposición de mercadería mediante Smart Replenishment

1.2 Problema

Actualmente para generar las recomendaciones para cada combinación de Tienda/Producto, el sistema Smart Replenishment utiliza una predicción de demanda basada en el promedio de ventas de los últimos 30 días. La metodología de predicción de demanda es la siguiente:

1. Se calcula el promedio de venta de los últimos 30 días.
2. Se multiplica la cantidad promedio diaria por la suma de los días de lead time y los días de stock a cubrir.

Aunque este enfoque es sencillo, se ha demostrado inexacto, generando sugerencias de envío de mercadería poco precisas. Esta falta de precisión afecta la eficiencia en la gestión de inventarios de las tiendas, conduciendo a problemas de sobrestock o desabastecimiento y por ende, una menor satisfacción del cliente (el fabricante) dado a que se perjudica su relación con las redes de supermercados en las que opera. La naturaleza dinámica del mercado retail exige modelos de predicción más robustos y precisos para este tipo de productos de consumo masivo, que puedan adaptarse a las fluctuaciones de la demanda en cada supermercado.

1.3 Objetivo

El objetivo principal de esta tesis es contribuir a la mejora de los sistemas de predicción de demanda en el sector retail, mediante el diseño y evaluación de una metodología basada en modelos de deep learning que combine precisión y escalabilidad operativa.

Con este propósito, se plantean dos objetivos específicos:

1. **Optimizar la predicción de demanda dentro del sistema Smart Replenishment**, mediante el desarrollo de una nueva estrategia de modelado capaz de superar el enfoque actual basado en promedios móviles. Se propone entrenar modelos a nivel de producto que integren simultáneamente la información de todas las tiendas asociadas, con el fin de captar patrones generales de comportamiento y mejorar la calidad de los pronósticos sin incurrir en altos costos computacionales.
2. **Desarrollar una metodología de modelado generalizable para la predicción de demanda a gran escala**, aplicable a entornos con miles de combinaciones producto-tienda. Esta metodología apunta a establecer una solución robusta y reutilizable que mantenga altos niveles de precisión y eficiencia, aún frente a escenarios crecientes en complejidad y volumen de datos.

El cumplimiento de estos objetivos permitirá no solo fortalecer el desempeño del sistema Smart Replenishment, sino también ofrecer una estrategia adaptable a otros desafíos similares en la industria del retail.

1.4 Estado del Arte

En la práctica actual, coexisten métodos estadísticos clásicos con técnicas avanzadas de machine learning y deep learning. Los enfoques tradicionales basados en la metodología Box–Jenkins (ARIMA), el suavizamiento exponencial y los modelos de espacio de estados continúan siendo ampliamente utilizados por su simplicidad, interpretabilidad y bajo costo computacional, y sirven habitualmente como benchmarks para evaluar propuestas más complejas (Hyndman & Athanasopoulos, 2021).

Por otro lado, los modelos de regresión basados en árboles—especialmente las versiones de boosting como XGBoost—se han consolidado como soluciones de referencia en entornos industriales de alta granularidad. XGBoost, presentado por Chen & Guestrin (2016), implementa un sistema de gradient boosting altamente escalable y regularizado, que ha demostrado un rendimiento sobresaliente en múltiples competiciones y aplicaciones reales (Chen & Guestrin, 2016).

En el ámbito del deep learning, se han desarrollado diferentes arquitecturas capaces de modelar dependencias temporales complejas. Las redes convolucionales aplicadas a series temporales, como las redes neuronales convolucionales (CNN) y sus combinaciones con redes neuronales recurrentes de Memoria a Corto Plazo (LSTM), han mostrado buenos resultados al capturar patrones espaciales y secuenciales en los datos (Borovykh et al., 2017). DeepAR, una red recurrente autorregresiva probabilística, aprovecha el entrenamiento conjunto sobre múltiples series relacionadas para generar intervalos de predicción confiables, mostrando mejoras de hasta un 15 % frente a técnicas tradicionales (Salinas et al., 2020). Por su parte, variantes inspiradas en WaveNet incorporan convoluciones dilatadas para modelar dependencias de largo alcance y se han utilizado con éxito en escenarios de predicción de alta frecuencia en retail (van den Oord et al., 2016).

Más recientemente, se han popularizado los Transformers adaptados a series temporales, como el Temporal Fusion Transformer (TFT), que integra mecanismos de atención para modelar relaciones complejas entre variables temporales y covariables exógenas como promociones, eventos o estacionalidad (Lim et al., 2021). Este tipo de modelos permite capturar dinámicas no lineales y relaciones jerárquicas de forma interpretable. En la misma línea, se han propuesto arquitecturas encoder-decoder con atención (Qin et al., 2017), y enfoques híbridos que combinan la Variational Mode Decomposition con redes de atención para descomponer la serie en múltiples componentes informativos antes del pronóstico (Wang et al., 2023).

Finalmente, a pesar de los avances en precisión, los modelos actuales siguen enfrentando desafíos en términos de escalabilidad, interpretabilidad y búsqueda eficiente de hiperparámetros. Por esta razón, en aplicaciones reales es frecuente el uso de estrategias de ensamblado que combinan modelos estadísticos, XGBoost y redes neuronales profundas para obtener soluciones robustas y balanceadas (Zhang et al., 2022).

2. Datos

2.1 Introducción a los Datos

Para alcanzar el objetivo de esta tesis, se utilizaron datos provistos por la empresa Scanntech. Se contó con información histórica de ventas de productos de las categorías “Refrigerantes” (todo tipo de bebidas no alcohólicas) y “Cervezas” de distintos supermercados ubicados en el departamento de São Paulo, Brasil.

A continuación, se detalla la información disponible en el dataset utilizado:

- pdv_codigo: Identificador del punto de venta. 30 valores únicos.
- codigo_barras_sku: Identificador del producto. 154 valores únicos.
- fecha_comercial: Identificador de la fecha de venta. 727 valores únicos, desde 2022-12-01 hasta 2024-11-30.
- nombre_sku: Nombre del producto. 154 valores únicos.
- cant_vta: *variable a predecir*. Cantidad de unidades vendidas.

Composición resultante del DataSet:

- 839.178 filas x 5 columnas.
- 1.319 combinaciones (pdv_codigo-codigo_barras_sku) que requieren predicción de demanda.

2.2 Análisis Exploratorio

Se realizó un análisis exploratorio de los datos con el objetivo de comprender mejor las características y patrones de venta de la información que se utilizó para el desarrollo de esta tesis.

A continuación se muestra el promedio de ventas según el día de la semana y el volumen de ventas por mes considerando todas las tiendas y puntos de venta disponibles.

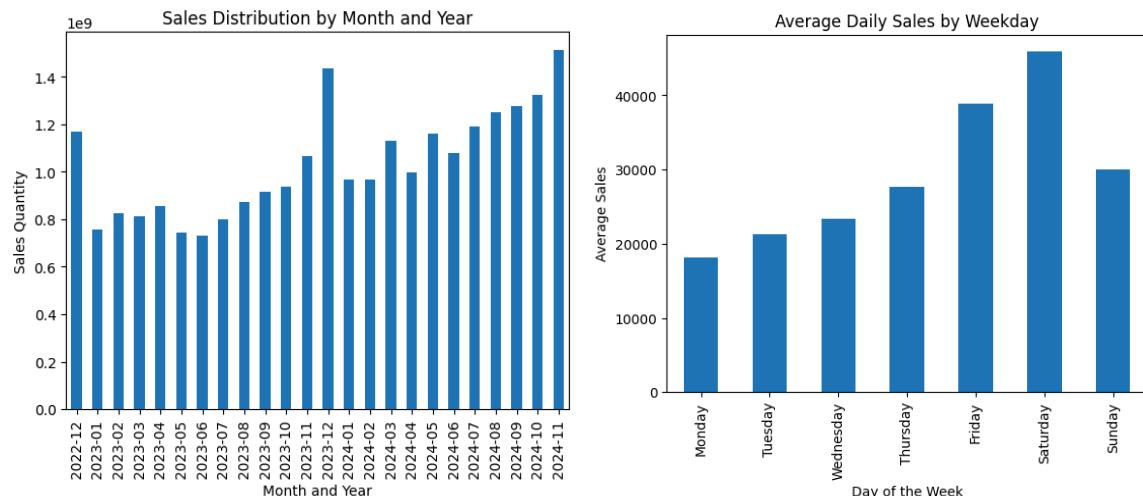


Figura 3. Distribución de ventas según mes y dia de la semana

Como se observa en los gráficos, existen dos patrones claros en el comportamiento de las ventas. En primer lugar, los días sábados presentan la mayor demanda semanal, lo que refleja un incremento en el consumo durante el fin de semana. Por otro lado, las ventas muestran un marcado crecimiento en los meses de verano, evidenciando una estacionalidad asociada a esta época del año. Estas tendencias sugieren que factores como el tiempo libre y las condiciones climáticas influyen directamente en el volumen de ventas de los productos analizados.

A continuación, se clasifican los puntos de venta según la cantidad de productos que comercializan, y los productos según la cantidad de puntos de venta en los que están disponibles.

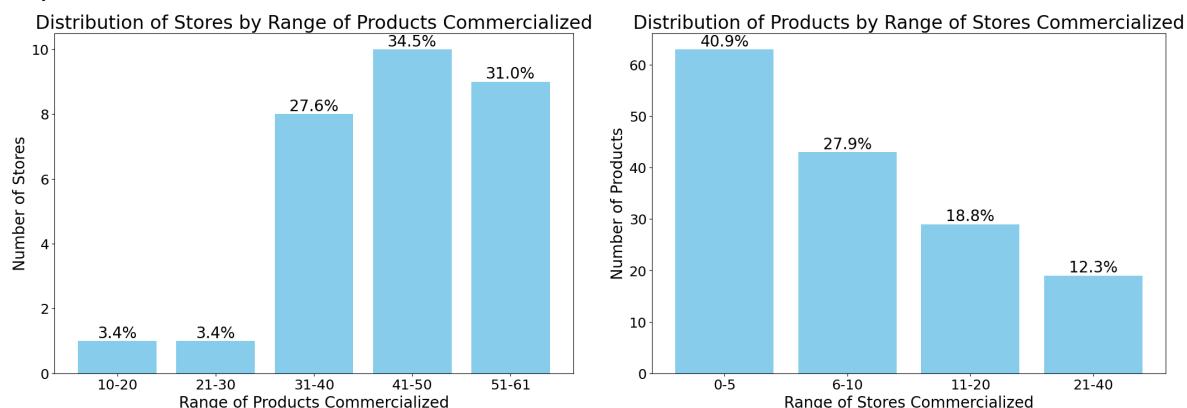


Figura 4. Distribución de tiendas y productos según su alcance comercial

El análisis de la figura izquierda muestra que la mayoría de las tiendas (93%) comercializan entre 31 y 61 productos (del total de 154 productos en los datos analizados), concentrándose principalmente en el rango de 41 a 50 productos (34.5%). Los puntos de venta con un surtido más limitado (10 a 30 productos) son muy pocos (6.8%).

Respecto a la figura de la derecha en relación a los productos, se observa que el 40.9% se comercializan en 5 o menos puntos de venta dentro de los datos analizados, lo que evidencia que la mayoría de los productos están disponibles en un número reducido de tiendas, mientras que una menor proporción se distribuye de manera más amplia.

En la siguiente imagen se pueden apreciar los promedios de venta diaria de los 10 productos más vendidos en los datos analizados.

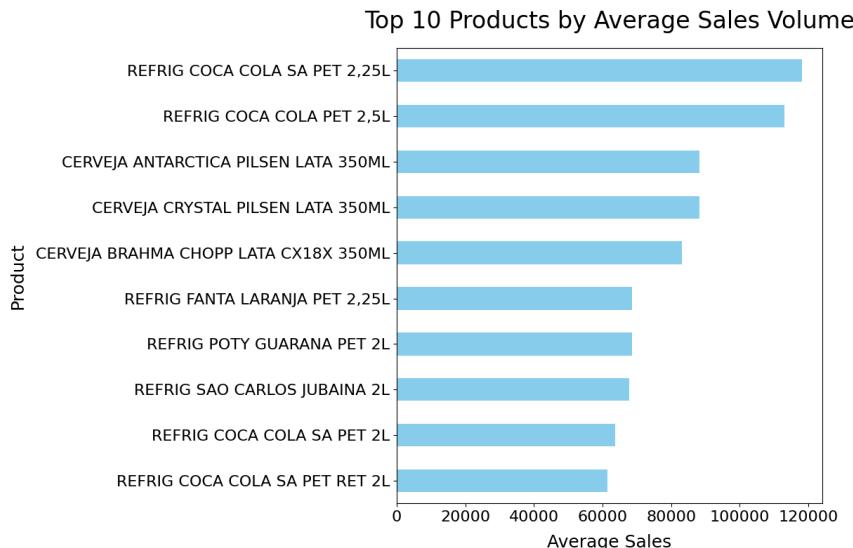


Figura 5. 10 productos con mayor volumen de venta

El estudio revela dos tendencias claras en el consumo: por un lado, se observa un alto volumen de ventas en los refrescos de Coca-Cola, destacando especialmente su presentación de 2.25 litros con un promedio diario de 117,841 unidades vendidas. Ocho de las diez referencias más comercializadas corresponden a esta marca, evidenciando su fuerte presencia en el mercado.

Por otro lado, las cervezas Antarctica y Brahma también muestran un importante nivel de ventas, ocupando posiciones relevantes entre los productos más demandados.

Estos resultados demuestran que ambos tipos de bebidas -refrescos y cervezas- mantienen un consumo elevado, aunque responden a preferencias y momentos de consumo distintos. El análisis confirma que estas dos categorías concentran la mayor demanda en el segmento de bebidas estudiado.

El siguiente gráfico muestra la distribución de la cantidad promedio de ventas totales diarias por punto de venta.

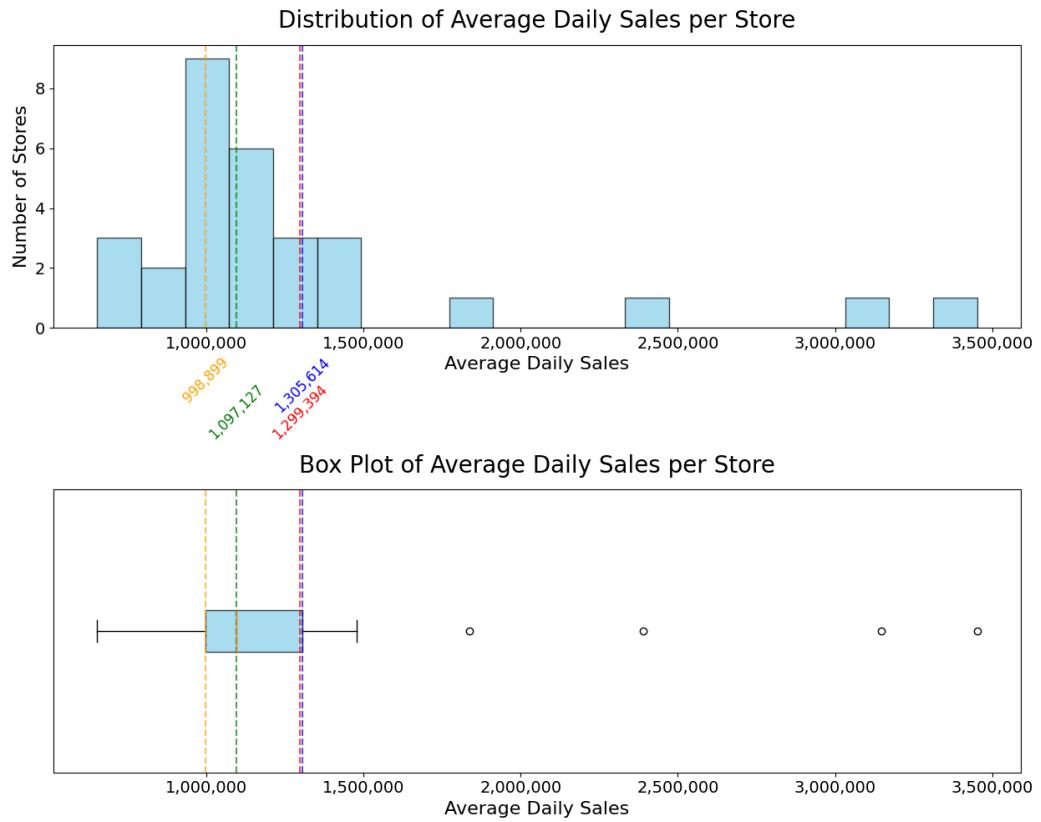


Figura 6. Distribución de la venta diaria promedio por tienda

El análisis de la distribución de las ventas promedio diarias por tienda muestra una concentración moderada alrededor de valores centrales. La media se ubica en aproximadamente 1.29 millones de unidades, mientras que la mediana es de 1.10 millones, lo que indica una ligera asimetría hacia tiendas con mayores ventas. El percentil 25 se sitúa en 998,899 unidades, lo que implica que el 25% de las tiendas tienen ventas promedio diarias por debajo de ese valor. Por otro lado, el percentil 75 alcanza los 1.31 millones, lo que indica que el 75% de las tiendas se encuentra por debajo de ese umbral. Esta distribución sugiere una variabilidad considerable entre tiendas, pero sin la presencia de una dispersión extrema en los valores centrales.

El siguiente gráfico muestra la distribución de la cantidad promedio de ventas totales diarias por producto.

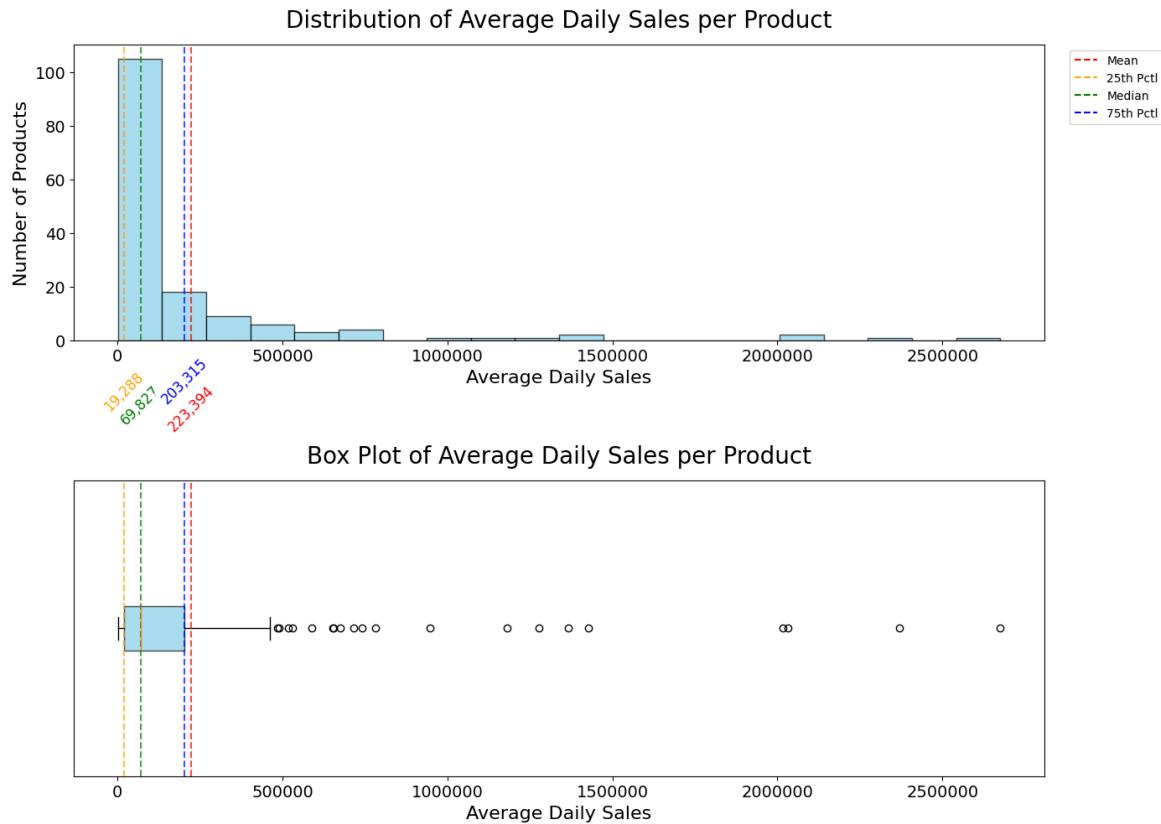


Figura 7. Distribución de la venta diaria promedio por producto

La distribución de las ventas promedio diarias por producto presenta una marcada dispersión, reflejo de diferencias significativas en el comportamiento de la demanda entre los distintos artículos. La media se sitúa en aproximadamente 223 mil unidades, mientras que la mediana —más representativa en distribuciones asimétricas— es considerablemente menor, con aproximadamente 69 mil unidades, lo que sugiere una concentración de productos con niveles de venta bajos y la presencia de algunos productos con volúmenes excepcionalmente altos. El percentil 25 se encuentra en 19 mil unidades, lo que indica que una cuarta parte de los productos tiene ventas diarias promedio por debajo de ese valor. Por su parte, el percentil 75 alcanza las 203 mil unidades aproximadamente, reflejando que el 75% de los productos se mantiene por debajo de este umbral. Esta distribución evidencia una fuerte asimetría positiva y destaca la necesidad de estrategias diferenciadas por tipo de producto en función de su volumen de ventas.

2.3 Clusterización

Dado que el enfoque propuesto entrena un único modelo por producto, se realizó una segmentación de los productos con el objetivo de analizar el desempeño de los modelos según distintos patrones de demanda.

Para ello, los productos fueron agrupados utilizando el algoritmo K-Means, empleando como variables sus perfiles de ventas promedio a nivel anual y trimestral. Estas métricas permiten capturar tanto tendencias generales como comportamientos estacionales característicos.

Los datos fueron previamente normalizados para asegurar una contribución equitativa de todas las variables durante el proceso de agrupamiento. La elección del número óptimo de clusters se realizó mediante el método del codo como se puede observar a continuación:

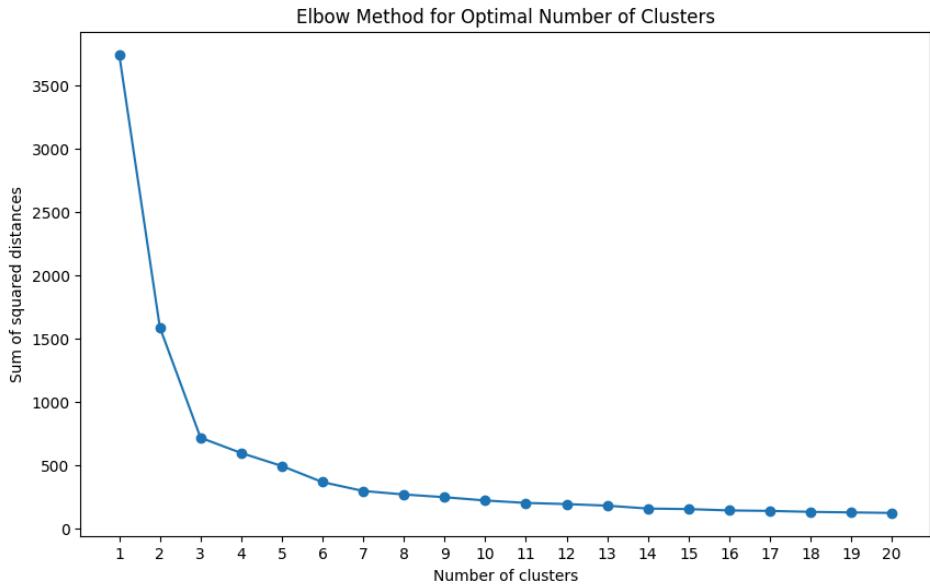


Figura 8. Método del codo para detectar número óptimo de clusters

A partir de este análisis, se seleccionó K=3 como el valor más adecuado, logrando una segmentación eficiente que permite agrupar productos con características de demanda similares.

Para analizar las diferencias entre los grupos obtenidos, se evaluó el promedio de ventas por trimestre en cada cluster. Esto permite identificar la estacionalidad y variaciones en la demanda dentro de cada grupo.

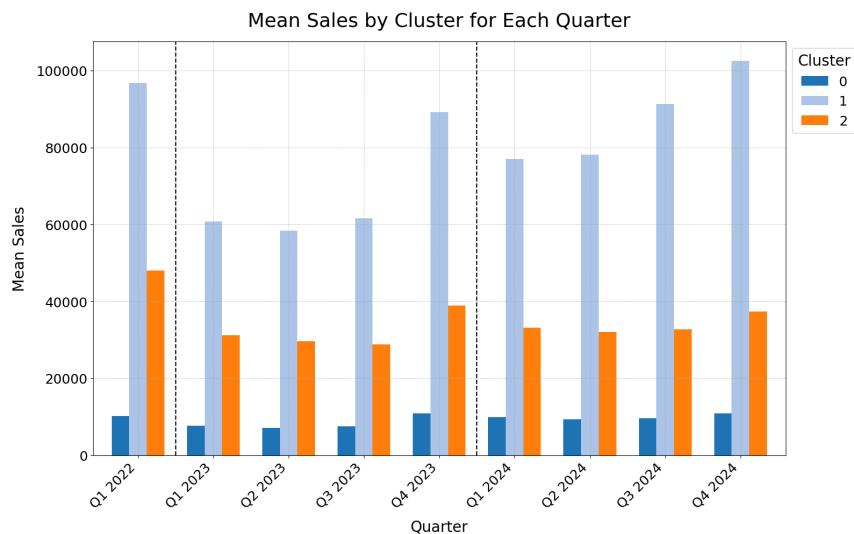


Figura 9. Venta promedio del trimestre según cluster

En la Figura se observa la evolución de las ventas promedio trimestrales por cluster a lo largo del tiempo. El Cluster 1 presenta consistentemente los niveles más altos de ventas, con marcadas variaciones estacionales, destacándose especialmente en los cuartos trimestres de cada año. El Cluster 2 muestra un comportamiento intermedio, con un patrón también estacional aunque menos pronunciado. Finalmente, el Cluster 0 agrupa productos con baja demanda, manteniendo ventas relativamente estables y considerablemente inferiores en comparación con los otros grupos.

También se analizó el promedio de ventas anuales por cluster, lo que evidencia diferencias en la magnitud y tendencia de la demanda a lo largo del tiempo.

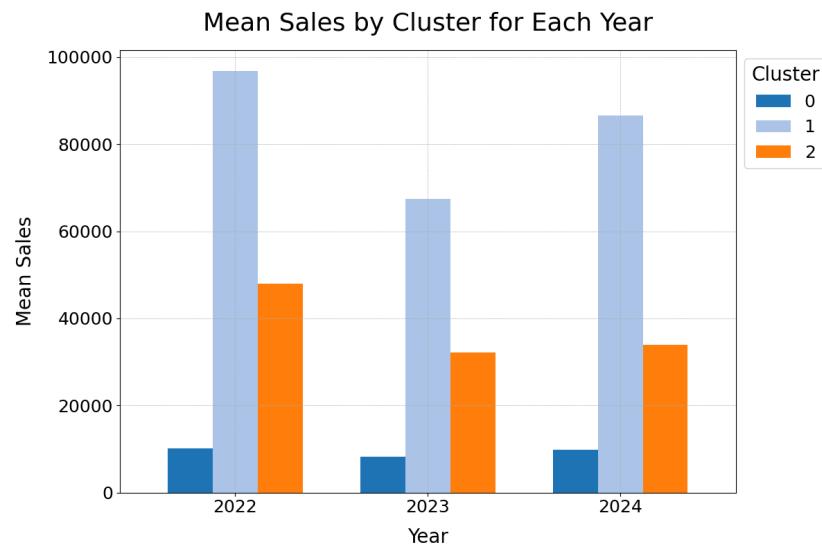


Figura 10. Venta promedio del año según cluster

El análisis del promedio de ventas anuales por cluster refuerza las diferencias en los niveles de demanda identificadas previamente. El Cluster 1 se mantiene como el grupo con mayor volumen de ventas, mostrando una leve recuperación en 2024 tras una caída en 2023. El Cluster 2 presenta una tendencia descendente desde 2022, lo que podría reflejar una pérdida de popularidad o estacionalidad decreciente en los productos que lo componen. Por su parte, el Cluster 0 mantiene niveles bajos de ventas a lo largo del período analizado, con variaciones mínimas entre años.

A continuación se presenta la distribución de los clusters, mostrando la cantidad de productos asignados a cada grupo. Esto proporciona una visión sobre la representatividad de cada segmento dentro del conjunto de datos.

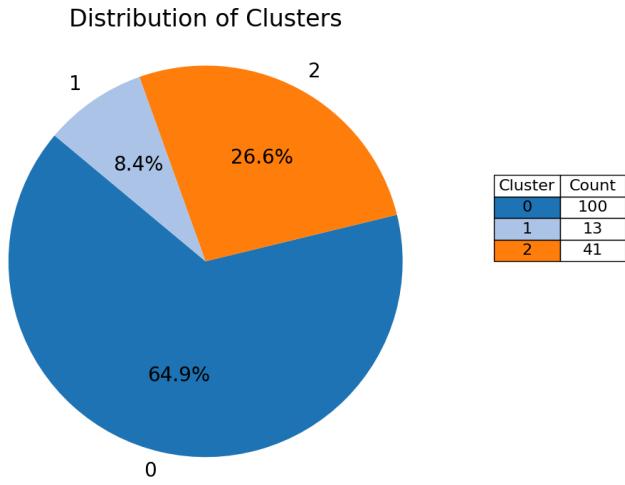


Figura 11. Distribución de productos según cluster

La distribución de productos entre clusters revela que el Cluster 0 concentra la mayoría de los artículos (64,9%), lo que sugiere que gran parte del catálogo comparte patrones de baja demanda. En contraste, el Cluster 1, aunque representa solo el 8,4% del total, agrupa los productos con mayores niveles de ventas, destacándose por su impacto en el volumen general. El Cluster 2, con un 26,6%, representa un segmento intermedio en cuanto a comportamiento de demanda.

Finalmente se presentan los resultados del análisis PCA (Análisis de Componentes Principales) de los clusters definidos.

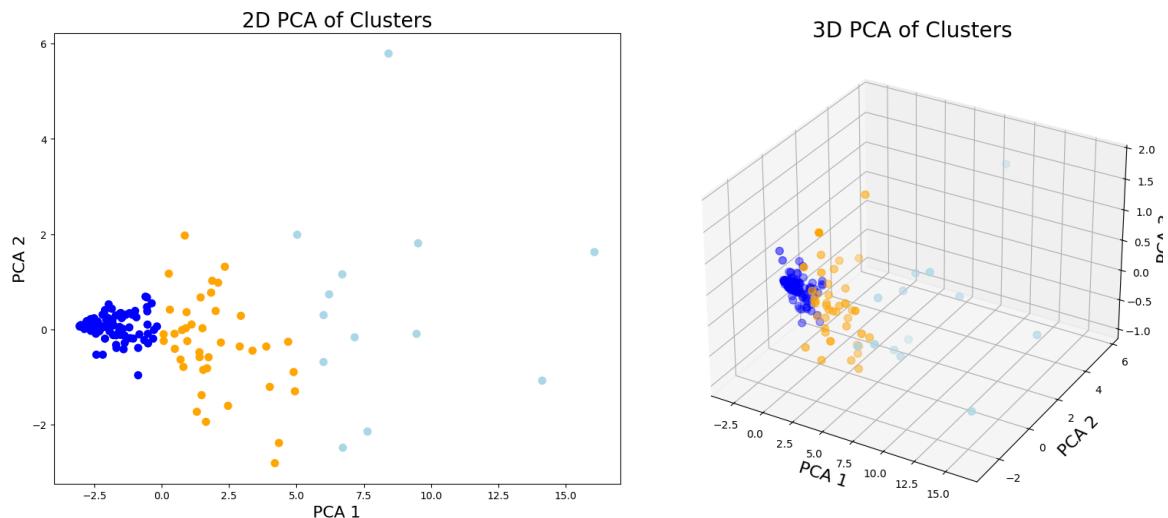


Figura 12. PCA análisis según dos y tres dimensiones

Se puede ver cómo, tanto definiendo dos como tres componentes principales, los productos muestran similitudes dentro de cada grupo y, al mismo tiempo, los tres grupos están bien separados entre sí según sus características de demanda.

2.4 Validación Problemática a Atacar

Para demostrar el problema que presenta la metodología de predicción de demanda del sistema de Smart Replenishment y la necesidad de incorporar modelos predictivos más precisos para este proceso se analizará el producto más vendido “REFRIG COCA COLA SA PET 2,25L” y el punto de venta 1 como referencia.

Supongamos que el día 2024-09-25, el sistema Smart Replenishment debe generar una sugerencia de envío de mercadería del producto analizado al punto de venta analizado. El punto de venta tiene configurado los siguientes parámetros:

- Día de emisión: 3 (representa al Miércoles)
- Día de entrega: 5 (representa al Viernes)
- Días stock: 18

Para determinar cuánta mercadería enviar, Smart Replenishment utiliza entre otras variables, una predicción de demanda del producto en el punto de venta desde el día de emisión del pedido (2024-09-25) hasta el último día a cubrir con el pedido (2024-10-15), es decir 20 días.

A continuación se muestra un gráfico que refleja el caso analizado:

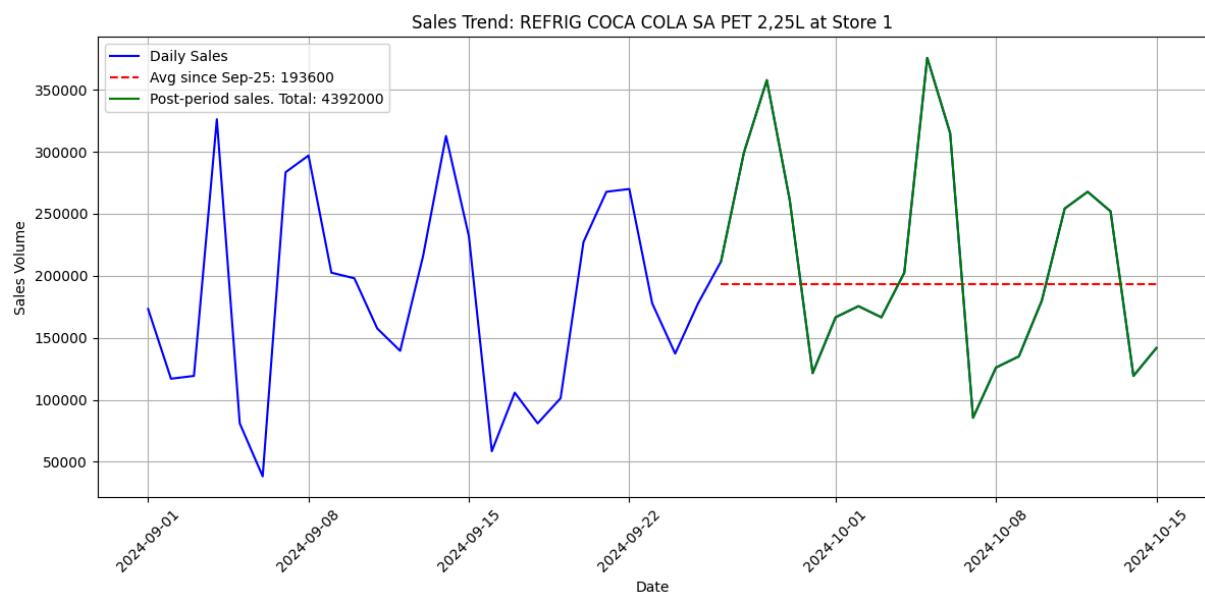


Figura 13. Análisis de demanda del producto y tienda particular

De esta forma el procedimiento de predicción de demanda actual es el siguiente:

1. Se calcula el promedio de venta de los últimos 30 días: 193.600 u / día
2. Se multiplica el promedio diario por la cantidad de días considerados: 20 días (2 días de Lead Time + 18 días de stock a cubrir) * 193.600 = 3.872.000 unidades.

Si solo consideramos la predicción de la demanda para generar la sugerencia, el sistema de Smart Replenishment sugeriría enviar 3.872.000 unidades siendo la demanda real de esos 20 días 4.392.000 unidades. Esto quiere decir que debido a la imprecisión en la

estimación de la demanda, el sistema sugirió enviar 520.000 unidades menos, generando pérdida de ventas y posibles rupturas de stock. Este problema se replica en las sugerencias de cada combinación de Tienda/Producto del sistema.

Lo que busca esta tesis es sustituir la metodología actualmente utilizada (línea punteada roja) por una predicción de ventas que se acerque lo más posible a lo que realmente ocurre (línea verde) aprendiendo de lo que ocurrió (línea azul).

3. Metodología

3.1 Modelos de Predicción Propuestos y Benchmarks

En esta sección se presentan los modelos benchmark utilizados para la evaluación comparativa y los modelos propuestos en esta investigación: Simple FeedForward (Multilayer Perceptron), DeepAR, Temporal Fusion Transformer (TFT) y WaveNet.

La selección de los modelos propuestos se realizó a partir de un proceso exploratorio en el cual se probaron distintos algoritmos de redes neuronales bajo una misma metodología de entrenamiento y evaluación. Esto permitió comparar de manera consistente el desempeño de cada enfoque. Se eligieron los cuatro modelos que mostraron mejor desempeño preliminar en precisión de pronóstico, antes de aplicar cualquier proceso de optimización de hiperparámetros. Esta estrategia permitió enfocar el trabajo en arquitecturas con mayor potencial de mejora y adecuadas para capturar patrones de demanda en el sector retail.

La elección de estos modelos también se respalda en el estado del arte de la predicción de series de tiempo con deep learning en el sector retail, donde estas arquitecturas han demostrado ser efectivas para modelar la complejidad y variabilidad de la demanda. Además, cada modelo aporta características técnicas específicas que los hacen especialmente apropiados para el problema abordado:

- **Simple FeedForward (SFF):** Captura relaciones no lineales de manera eficiente, destacándose por su simplicidad y rendimiento en la modelización de series de tiempo con patrones complejos.
- **DeepAR:** Permite capturar dinámicas temporales complejas mediante la modelización probabilística de series de tiempo, siendo ideal para entornos con alta variabilidad de demanda.
- **Temporal Fusion Transformer (TFT):** Integra mecanismos de atención e interpretabilidad que facilitan la captura de relaciones multivariadas y patrones a largo plazo.
- **WaveNet:** Utiliza convoluciones dilatadas para modelar de forma eficiente dependencias temporales de corto y largo alcance, optimizando el rendimiento sin necesidad de arquitecturas recurrentes.

En conjunto, los modelos propuestos permiten abordar la predicción de demanda combinando diferentes enfoques de modelado de series de tiempo, maximizando así las posibilidades de obtener resultados precisos y robustos.

3.1.1 Modelos Benchmark

Para evaluar el desempeño de los modelos propuestos, se definieron dos benchmarks de comparación:

Benchmark actual: Venta media de los últimos 30 días

El primer modelo benchmark replica el enfoque actualmente utilizado por el sistema Smart Replenishment, que predice la demanda tomando el promedio de ventas de los últimos 30 días para cada producto en cada tienda. Este valor promedio se emplea como pronóstico fijo para todos los días futuros. Si bien este método es simple y rápido de implementar, presenta limitaciones importantes, ya que no captura patrones temporales, tendencias ni eventos específicos que afectan las ventas reales. Su propósito en esta investigación es servir como base de comparación para demostrar cómo modelos más avanzados pueden mejorar la precisión de las predicciones al modelar de manera más realista la dinámica de la demanda en el sector retail.

Benchmark adicional: XGBoost

El segundo modelo benchmark seleccionado fue XGBoost, un algoritmo ampliamente utilizado en la industria para resolver problemas de predicción, dada su alta capacidad para modelar relaciones complejas y su eficiencia en tareas de regresión y clasificación. La inclusión de XGBoost como benchmark adicional permitió validar no solo la mejora respecto al enfoque tradicional, sino también evaluar el desempeño de los modelos propuestos frente a una técnica de referencia robusta y moderna. Una vez detectado el mejor modelo de los propuestos, se realizó una comparación con XGBoost para evaluar su desempeño en relación con una técnica de vanguardia. La comparación entre modelos se llevó a cabo siguiendo la misma metodología de entrenamiento, validación y evaluación de resultados, garantizando así la equidad en el análisis de desempeño.

3.1.2 Simple FeedForward: Multilayer Perceptron

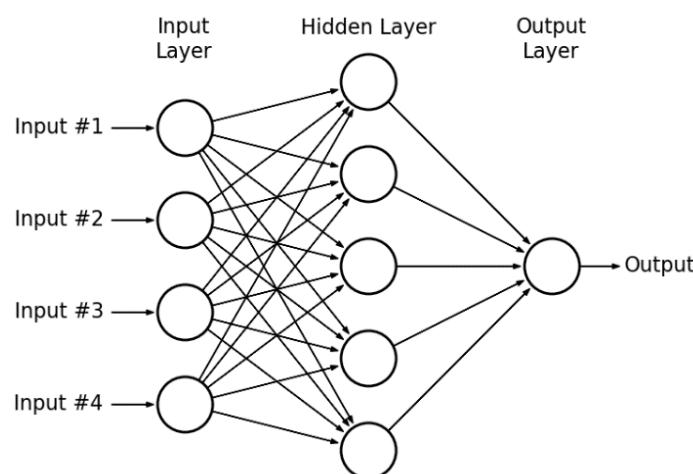


Figura 14. Representación del modelo de Multilayer Perceptron

3.1.2.1 Fundamento Teórico

El modelo Simple Feedforward, también conocido como Multilayer Perceptron (MLP), pertenece a la familia de redes neuronales densamente conectadas. Cada neurona de una capa está conectada con todas las de la siguiente, sin retroalimentación ni memoria, lo que hace que el modelo procese los datos de forma puramente estática (Bishop, 1995). Gracias a sus funciones de activación no lineales, el MLP es capaz de modelar relaciones no lineales entre variables. En el contexto de series temporales, el MLP utiliza ventanas deslizantes del pasado reciente como entrada para predecir el futuro. Si bien no captura explícitamente las dependencias temporales de largo plazo como otros modelos secuenciales, su simplicidad estructural y bajo costo computacional lo hacen adecuado para series con patrones lineales, estacionalidades simples o cuando se requiere una solución rápida y efectiva.

3.1.2.2 Arquitectura y Funcionamiento

- Entrada: El modelo recibe una ventana de contexto de longitud fija (context length) que representa los valores históricos recientes de la serie temporal.
- Componentes del modelo: La red está compuesta por una o más capas densas (hidden dimensions). Cada neurona está completamente conectada con las neuronas de la capa anterior y posterior, permitiendo al modelo capturar relaciones no lineales en los datos.
- Salida: El modelo genera una predicción para el horizonte futuro (prediction length). Dependiendo de la configuración, puede producir:
 - Predicción puntual: una única estimación del valor futuro.
 - Predicción probabilística: una distribución sobre los posibles valores futuros, comúnmente mediante StudentTOutput.
- Función de pérdida: Se utiliza la log-verosimilitud negativa (Negative Log-Likelihood) basada en la distribución de salida elegida. Esto permite al modelo aprender de manera probabilística la forma de los datos futuros.

3.1.2.3 Implementación en GluonTS

En este proyecto, se implementó el modelo Simple Feedforward utilizando el SimpleFeedForwardEstimator proporcionado por la biblioteca GluonTS (GluonTS, s.f.).

Parámetros clave del estimador:

- context length: número de pasos temporales observados utilizados como entrada para realizar la predicción.
- prediction length: horizonte de predicción, es decir, el número de pasos futuros que se desea pronosticar.
- hidden dimensions: estructura de la red oculta, definida como una lista que indica el número de neuronas por capa oculta (por ejemplo, [40, 40]).
- distribution output: tipo de distribución utilizada para generar predicciones probabilísticas (por ejemplo, Gaussian o StudentT).
- batch normalization: aplicación de normalización por lotes después de cada capa oculta, lo cual mejora la estabilidad y la velocidad del entrenamiento.

Entrenamiento con PyTorch Lightning: El proceso de entrenamiento se realiza a través de PyTorch Lightning, lo cual permite una integración clara y controlada de los siguientes hiperparámetros:

- learning rate: tasa de aprendizaje utilizada por el optimizador.
- weight decay: coeficiente de regularización L2 para evitar sobreajuste.
- batch size: número de muestras utilizadas en cada lote de entrenamiento.
- number of batches per epoch: cantidad de lotes procesados en cada época.
- maximum epochs: número máximo de épocas para el entrenamiento (en este trabajo, se fijó en 40).

3.1.2.4 Ventajas para el Retail

- Simplicidad: Fácil de implementar y ajustar, lo que permite una rápida experimentación.
- Eficiencia computacional: Requiere menos recursos en comparación con modelos más complejos como RNNs o Transformers, siendo adecuado para productos con demanda estable.
- Flexibilidad: Capacidad para adaptarse a diferentes configuraciones y necesidades del negocio, permitiendo su integración en sistemas de predicción a gran escala.

3.1.3 DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks

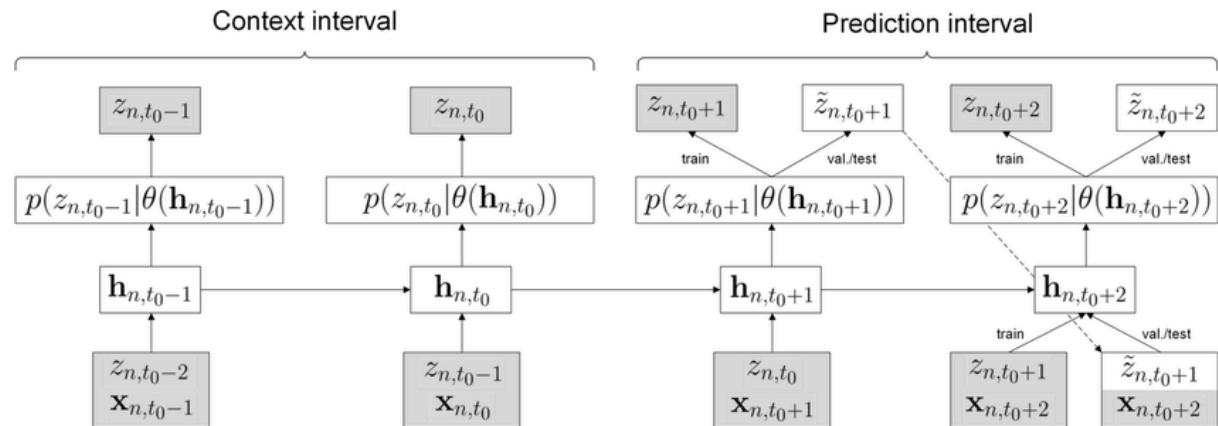


Figura 15. Representación del modelo DeepAR

3.1.3.1 Fundamento Teórico

DeepAR es un modelo de pronóstico probabilístico autorregresivo basado en redes neuronales recurrentes (RNNs), típicamente Long Short-Term Memory (LSTM) o Gated Recurrent Units (GRU), introducido por Salinas et al. Este diseño le permite capturar patrones temporales tanto a corto como a largo plazo en múltiples series correlacionadas, aprovechando la información compartida entre ellas para mejorar las predicciones individuales, especialmente en series con datos limitados. Su efectividad se destaca en contextos como la predicción de demanda de productos, donde existen similitudes entre series individuales. El modelo aprende una distribución condicional de la demanda futura basada en la historia pasada, utilizando el mecanismo de teacher forcing durante el

entrenamiento. En este proceso, el modelo recibe como entrada el valor real observado en lugar de su propia predicción anterior, lo que mejora la estabilidad del aprendizaje secuencial. Durante la inferencia, el modelo opera de manera autorregresiva, utilizando la predicción de un paso de tiempo como entrada para predecir el siguiente. Además, su capacidad para modelar explícitamente la incertidumbre mediante distribuciones paramétricas (como Student-t o Binomial Negativa) le permite cuantificar la incertidumbre y generar intervalos de confianza robustos, lo cual es crucial en entornos con alta variabilidad (Salinas et al., 2020).

3.1.3.2 Arquitectura y Funcionamiento

- Entrada: El modelo recibe una secuencia temporal de longitud fija (context length) y puede incorporar variables dinámicas (como calendarios o promociones) y estáticas (como el tipo de tienda o producto).
- Componentes del modelo: DeepAR está basado en redes recurrentes (RNNs), específicamente LSTM o GRU. Estas redes procesan secuencias para capturar dependencias temporales de corto y largo plazo.
Cada serie se escala individualmente (media y desviación estándar) para estabilizar el entrenamiento y facilitar el aprendizaje.
- Salida: El modelo estima una distribución probabilística sobre los valores futuros (prediction length). Esta distribución captura la incertidumbre inherente en las predicciones.
- Función de pérdida: Se utiliza la log-verosimilitud negativa, comparando la distribución generada con los valores reales observados, lo que permite un entrenamiento probabilístico coherente.

3.1.3.3 Implementación en GluonTS

En este proyecto, se implementó el modelo DeepAR utilizando el DeepAREstimator de la biblioteca GluonTS (GluonTS, s.f.).

Parámetros clave del estimador:

- context length: número de pasos anteriores utilizados como entrada del modelo.
- prediction length: horizonte de predicción a futuro.
- number of layers: cantidad de capas recurrentes en la arquitectura del modelo.
- hidden size: dimensión del estado oculto en cada capa LSTM.
- dropout rate: tasa de abandono aplicada entre capas para reducir el riesgo de sobreajuste.
- batch size: número de muestras por lote durante el entrenamiento.
- frequency: frecuencia temporal de los datos (por ejemplo, diaria, semanal).
- number of parallel samples: número de trayectorias de muestra generadas durante la inferencia.

Entrenamiento con PyTorch Lightning: El modelo se entrenó utilizando PyTorch Lightning, lo que permite una gestión modular y reproducible del entrenamiento. Se configuraron los siguientes hiperparámetros:

- learning rate: tasa de aprendizaje.
- weight decay: coeficiente de regularización L2.

- maximum epochs: número máximo de épocas de entrenamiento (establecido en 40).

3.1.3.4 Ventajas para el Retail

DeepAR presenta varias ventajas en el contexto del retail:

- Escalabilidad: Puede entrenar eficientemente sobre miles de series en paralelo (por ejemplo, un producto en múltiples tiendas), compartiendo parámetros.
- Manejo de datos escasos: Presenta robustez ante ventas intermitentes o con muchos ceros, especialmente al utilizar una distribución adecuada como la de Student-t.
- Integración de covariables: Permite la incorporación de información externa relevante (por ejemplo, calendario local, eventos especiales), lo cual es fundamental en escenarios reales de retail.
- Predicciones probabilísticas: Genera predicciones con múltiples valores posibles para cada punto en el tiempo, permitiendo estimar intervalos de confianza y tomar decisiones más informadas.

En este trabajo, se consideraron tanto la media como la mediana de las predicciones generadas por DeepAR, seleccionando aquella que presentó el menor error RMSE para cada combinación específica de producto y tienda, garantizando así las estimaciones más precisas para cada caso particular.

3.1.4 Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting

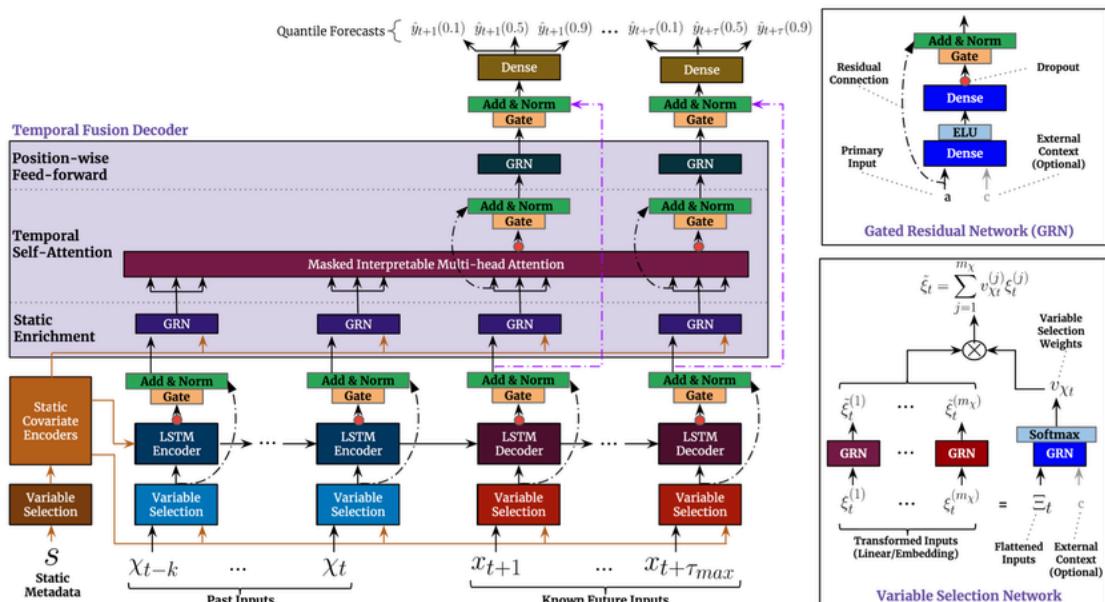


Figura 16. Representación del modelo Temporal Fusion Transformer

3.1.4.1 Fundamento Teórico

El Temporal Fusion Transformer (TFT) es un modelo diseñado para el pronóstico multivariado y de múltiple horizonte de series temporales, combinando mecanismos de atención con redes recurrentes. Su arquitectura híbrida incorpora capas LSTM para capturar patrones locales y mecanismos de autoatención para modelar relaciones a largo plazo entre las entradas temporales. Además, el TFT se distingue por su capacidad para seleccionar automáticamente las variables más relevantes mediante mecanismos de compuertas, lo que le otorga una ventaja en entornos con múltiples covariables. Esta arquitectura no solo permite obtener predicciones precisas, sino también interpretables, ofreciendo información sobre la importancia relativa de cada entrada. Este aspecto es fundamental en aplicaciones como retail o finanzas, donde la trazabilidad del modelo es crítica (Lim et al., 2021).

3.1.4.2 Arquitectura y Funcionamiento

- Entrada: El modelo recibe una secuencia de longitud fija (context length) que incluye valores históricos de la variable objetivo, variables dinámicas (como calendarios, promociones o días festivos) y variables estáticas (como tipo de tienda o familia de producto).
- Componentes del modelo: TFT integra varios bloques especializados:
 - Codificación temporal: capas LSTM extraen patrones secuenciales.
 - Autoatención temporal: permite identificar relaciones entre diferentes momentos en la serie.
 - Mecanismos de compuertas: redes de selección de variables y decodificadores de fusión que integran información estática y dinámica de forma adaptativa.
- Salida: El modelo genera múltiples cuantiles de la distribución futura sobre un horizonte de predicción (prediction length), típicamente P10, P20, ..., P90. Esto permite obtener una predicción central (como la mediana P50) acompañada de bandas de confianza, facilitando la interpretación de la incertidumbre en los resultados.
- Función de pérdida: Se utiliza la Quantile Loss, que optimiza simultáneamente varios cuantiles. Esto permite generar intervalos de predicción confiables y robustos frente a valores atípicos.

3.1.4.3 Implementación en GluonTS

En este proyecto, se utilizó el `TemporalFusionTransformerEstimator` de la biblioteca GluonTS (GluonTS, s.f.).

Parámetros clave del estimador:

- `context length`: número de pasos anteriores utilizados como entrada del modelo.
- `prediction length`: longitud del horizonte de predicción.
- `hidden dimension`: dimensión de las capas ocultas internas del modelo.
- `variable dimension`: dimensión de los embeddings para las variables de entrada.
- `number heads`: cantidad de cabezas utilizadas en el mecanismo de atención multi-cabeza.
- `dropout rate`: tasa de abandono utilizada para regularización entre capas.
- `batch size`: tamaño del lote de datos utilizado durante el entrenamiento.
- `frequency`: frecuencia temporal de los datos.

Entrenamiento con PyTorch Lightning: El modelo fue entrenado utilizando PyTorch Lightning, lo que permite una gestión estructurada del ciclo de entrenamiento. Los hiperparámetros relevantes definidos fueron:

- learning rate: tasa de aprendizaje.
- weight decay: coeficiente de regularización L2.
- maximum epochs: número máximo de épocas de entrenamiento (configurado en 40).

3.1.4.4 Ventajas para el Retail

El TFT ofrece varias ventajas que lo hacen especialmente adecuado para aplicaciones en el sector retail:

- Interpretabilidad: Permite identificar los factores clave que impulsan la demanda, como promociones o eventos específicos, lo que facilita la toma de decisiones estratégicas.
- Flexibilidad con covariables: Maneja eficazmente combinaciones de variables estáticas (por ejemplo, región de la tienda) y dinámicas (por ejemplo, clima, promociones), adaptándose a diferentes contextos y productos.
- Alto rendimiento en múltiples series: Aprovecha las correlaciones entre diferentes productos o tiendas, mejorando la precisión en escenarios complejos y permitiendo escalar a grandes volúmenes de datos.

3.1.5 WaveNet: A Generative Model for Raw Audio

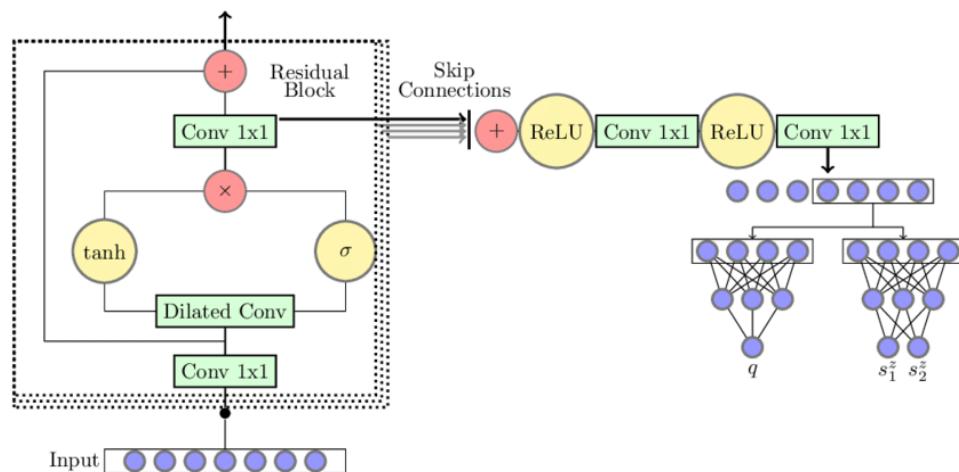


Figura 17. Representación del modelo WaveNet

3.1.5.1 Fundamento Teórico

WaveNet es una red neuronal convolucional profunda originalmente desarrollada para la síntesis de audio, pero posteriormente adaptada al pronóstico de series temporales. Su arquitectura se basa en convoluciones dilatadas causales, lo que le permite capturar dependencias a largo plazo sin violar la causalidad temporal (es decir, sin utilizar información futura). A diferencia de los modelos secuenciales autoregresivos tradicionales, WaveNet ofrece la ventaja de una mayor parallelización durante el entrenamiento y la inferencia, lo que mejora la eficiencia computacional. Su diseño modular y la capacidad de

generar salidas probabilísticas lo convierten en una alternativa robusta para modelar series con alta irregularidad, estacionalidades complejas o comportamientos no lineales, escenarios comunes en la predicción de demanda en retail (Van den Oord et al., 2016).

3.1.5.2 Arquitectura y Funcionamiento

- Entrada: El modelo recibe secuencias temporales de longitud fija que representan los valores históricos recientes. La longitud de esta ventana de entrada no se define manualmente, sino que se calcula automáticamente a partir de la estructura del modelo (cantidad de capas y profundidad de los filtros), asegurando que tenga en cuenta suficiente historial para realizar predicciones. Además, pueden incluirse variables categóricas estáticas, como el identificador de tienda o producto.
- Componentes del modelo: WaveNet emplea bloques convolucionales con varias características:
 - Capas convolucionales dilatadas: permiten al modelo detectar patrones a distintas escalas temporales, ampliando progresivamente la cantidad de pasos del pasado que puede observar.
 - Enmascaramiento causal: asegura que las predicciones en un momento dado solo utilicen información de momentos anteriores, respetando el orden temporal.
 - Conexiones residuales: facilitan el entrenamiento estable de redes profundas, reduciendo problemas como la pérdida de información durante el aprendizaje.
- Salida: El modelo genera una distribución probabilística para cada paso del horizonte de predicción (prediction length), estimando la cantidad de unidades a vender por día. Dado que se trata de variables discretas y no negativas (como ventas diarias), se utiliza una distribución binomial negativa, que permite capturar tanto la media esperada como la variabilidad en la demanda.
- Función de pérdida: Se utiliza la log-verosimilitud negativa (Negative Log-Likelihood), maximizando la probabilidad de los datos observados bajo la distribución predicha.

3.1.5.3 Implementación en GluonTS

Para este proyecto, se empleó el WaveNetEstimator de la biblioteca GluonTS (GluonTS, s.f.).

Parámetros clave del estimador:

- prediction length: horizonte de predicción.
- frequency: frecuencia temporal de los datos.
- number residual channels: número de canales en las capas residuales convolucionales.
- number skip channels: número de canales utilizados para las conexiones de salto (skip connections).
- number stacks: cantidad de veces que se repite la pila de capas dilatadas, lo cual afecta la profundidad efectiva del modelo.
- number bins: cantidad de bins utilizados para la discretización de la salida.
- embedding dimension: dimensión de los embeddings para variables categóricas estáticas.

Entrenamiento con PyTorch Lightning: El entrenamiento se llevó a cabo con PyTorch Lightning, lo que permite controlar eficientemente el proceso de optimización. Los hiperparámetros asociados fueron:

- learning rate: tasa de aprendizaje.
- weight decay: coeficiente de regularización L2.
- batch size: tamaño del lote para cada paso de entrenamiento.
- number batches per epoch: número de lotes por época utilizados para simular un epoch sobre un subconjunto de los datos.
- maximum epochs: número máximo de épocas (configurado en 40).
- gradient clip value: valor máximo permitido para el gradiente, utilizado para estabilizar el entrenamiento (configurado en 0.1 - umbral comúnmente efectivo para prevenir explosiones del gradiente sin restringir excesivamente la magnitud de las actualizaciones).

3.1.5.4 Ventajas para el Retail

WaveNet aporta beneficios específicos para aplicaciones de demanda en el sector retail:

- Captura de patrones multiescala: puede identificar simultáneamente tendencias diarias, semanales o estacionales, cruciales para la planificación de inventario y promociones.
- Robustez ante outliers: su naturaleza convolucional atenúa el impacto de ruidos o valores atípicos en los datos históricos de ventas.
- Predicciones precisas y rápidas: gracias a la paralelización, ofrece eficiencia computacional sin sacrificar calidad predictiva.

Aunque WaveNet fue concebido inicialmente para la generación de audio, su arquitectura basada en convoluciones dilatadas causales ha demostrado ser eficaz en la predicción de series temporales en el ámbito del retail. Estudios como el de Borovskykh et al. (2017) han adaptado WaveNet para el pronóstico de series financieras, evidenciando su versatilidad y capacidad para capturar dependencias temporales complejas. En este trabajo, se empleó WaveNet para predecir la demanda de productos, considerando tanto la media como la mediana de las distribuciones generadas. Para cada combinación específica de producto y tienda, se seleccionó la estimación que minimiza el error RMSE, asegurando así la mayor precisión posible en las predicciones.

3.2. Metodología de Entrenamiento y Validación

3.2.1 Definición de Conjuntos de Entrenamiento, Validación y Prueba

Dado que el objetivo del modelo es predecir la demanda diaria a 30 días para cada combinación producto-tienda, se implementó una estrategia de validación temporal que garantiza una evaluación realista del desempeño predictivo. Esta metodología preserva la estructura cronológica de los datos, evitando el filtrado de información y asegurando que las métricas reflejen el comportamiento del modelo en condiciones operativas reales.

Partición de Datos:

El conjunto completo (desde diciembre de 2022 hasta noviembre de 2024) se dividió de la siguiente manera:

- Entrenamiento (Train):
 - Periodo: 1 de diciembre de 2022 al 31 de octubre de 2024.
 - Justificación: Proporciona suficiente historia (casi dos años) para capturar patrones estacionales, tendencias y eventos relevantes.
- Validación (Validation):
 - Periodo: Últimos 30 días del conjunto de entrenamiento (1 al 31 de octubre de 2024).
 - Propósito: Optimización de hiperparámetros y selección del mejor modelo sin usar datos futuros.
- Prueba (Test):
 - Periodo: 1 al 30 de noviembre de 2024 (30 días, mismo horizonte de pronóstico requerido).
 - Propósito: Evalúa el desempeño final en datos nunca vistos por el modelo, simulando un escenario real de implementación.

3.2.2 Entrenamiento por Producto

Para el entrenamiento de los modelos, se adoptó un enfoque basado en la segmentación por producto. Es decir, para cada producto, se consolidaron todas las series temporales correspondientes a sus ventas en las distintas tiendas en las que está presente. Una vez generado este conjunto de datos, el modelo correspondiente al producto se entrena utilizando todas las series de manera conjunta. De forma análoga, en la etapa de predicción, el modelo genera pronósticos simultáneamente para todas las tiendas en las que el producto participa.

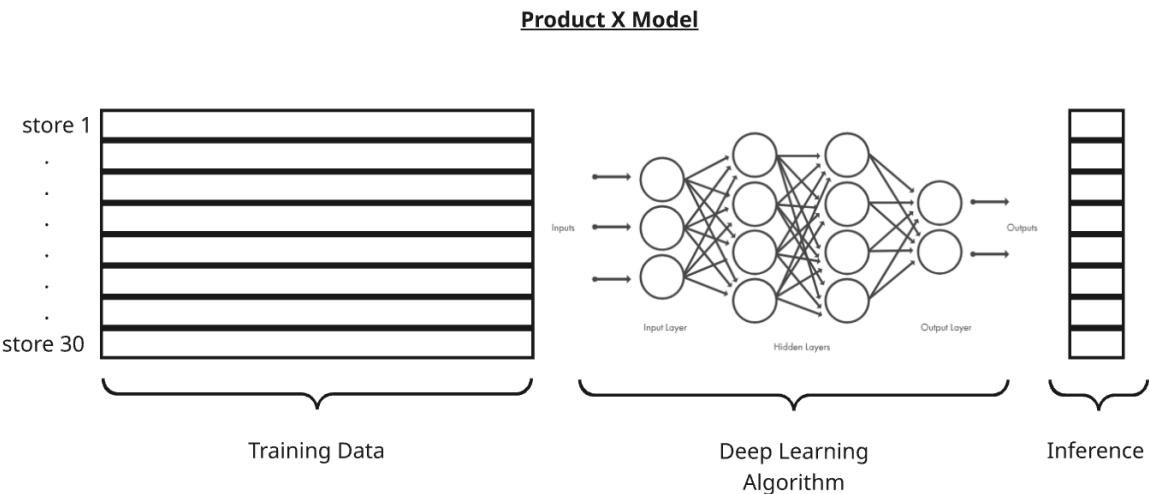


Figura 18. Metodología de entrenamiento por producto

Este enfoque se justificó por varias razones. En primer lugar, entrenar un modelo independiente para cada combinación de producto y tienda habría resultado computacionalmente costoso e ineficiente, tanto en recursos como en tiempos de entrenamiento, debido al elevado número de combinaciones posibles. En cambio, al agrupar todas las series correspondientes a un mismo producto en un único modelo, se redujo significativamente el tiempo necesario para entrenar los modelos y se mejoró la escalabilidad del proceso. Además, este esquema permitió al modelo aprender patrones y tendencias generales que trascienden tiendas individuales, favoreciendo una mejor capacidad de generalización. También se aprovechó la información disponible en tiendas con históricos más extensos, lo que benefició la predicción en aquellas con series más cortas o ruidosas.

Este procedimiento fue aplicado de manera uniforme a todos los modelos considerados en este trabajo, garantizando así una base de comparación consistente entre ellos.

3.3. Metodología de Optimización de Modelos

3.3.1 Metodología de Selección de Hiperparámetros

En el desarrollo de modelos predictivos para series temporales, la selección de hiperparámetros representa una etapa crítica para alcanzar el máximo rendimiento de los algoritmos. En este estudio se desarrollaron dos enfoques: inicialmente se implementó Random Search como aproximación exploratoria, y posteriormente Bayesian Optimization como método más refinado, el cual se adoptó para los experimentos finales debido a sus ventajas observadas en las pruebas comparativas que se comentarán más adelante en el presente trabajo.

3.3.1.1 Estrategia de Validación y selección

La evaluación del desempeño se realizó sobre el conjunto de validación, conformado por los últimos 30 días del conjunto de entrenamiento, utilizando como métrica principal el Error Cuadrático Medio Raíz (RMSE). Este RMSE se calculó de forma agregada, combinando los errores de predicción de todas las tiendas asociadas al modelo correspondiente a cada producto, lo cual permitió obtener una medida unificada de precisión por producto.

Durante la búsqueda de hiperparámetros, se ejecutaron 10 iteraciones por modelo, balanceando el costo computacional con la calidad esperada de los resultados. Para definir el espacio de búsqueda, cada hiperparámetro fue clasificado según su tipo: categórico, entero o continuo. Esta clasificación permitió aplicar un esquema de muestreo diferenciado:

- Para hiperparámetros categóricos (por ejemplo, número de capas o dimensiones de embedding), se empleó una selección uniforme entre opciones predefinidas.
- Los hiperparámetros continuos se muestran de manera uniforme dentro de rangos acotados.
- En el caso de hiperparámetros sensibles a órdenes de magnitud (como la tasa de aprendizaje o el weight decay), se utilizó un muestreo log-uniforme, facilitando una exploración más eficiente en escalas exponenciales.

Este enfoque adaptativo permitió explorar de forma efectiva el espacio de configuraciones, favoreciendo la identificación de combinaciones robustas con un número de evaluaciones limitado.

Una vez seleccionada la mejor configuración por producto —es decir, aquella que obtuvo el menor RMSE en validación—, se almacenó información detallada sobre cada modelo: hiperparámetros óptimos, número de prueba en que fueron encontrados, cantidad de épocas entrenadas, y evolución de la pérdida tanto en entrenamiento como en validación.

El registro sistemático de esta información resultó clave para analizar el comportamiento de los modelos durante el entrenamiento, ajustar el espacio de búsqueda en función de los resultados y determinar de forma empírica una cantidad máxima de épocas adecuada antes de que el modelo dejara de mejorar. Además, este procedimiento aportó mayor control al proceso, facilitó comparaciones estructuradas entre modelos y productos, y reforzó la trazabilidad y transparencia metodológica del experimento.

Una vez finalizada la etapa de validación, se procedió al reentrenamiento de cada modelo utilizando la totalidad de los datos históricos disponibles, es decir, combinando los subconjuntos de entrenamiento y validación originales. En esta fase, se respetaron tanto los hiperparámetros seleccionados como la cantidad de épocas correspondiente al mejor desempeño registrado.

Finalmente, la evaluación se realizó sobre un conjunto de prueba completamente aislado durante todo el proceso de ajuste y optimización, asegurando una medición imparcial del desempeño general.

3.3.1.2 Random Search: Exploración Eficiente del Espacio de Parámetros

Como primer enfoque se implementó la estrategia de Random Search, siguiendo las recomendaciones de Bergstra y Bengio (2012), quienes demostraron su eficacia frente a métodos como Grid Search en espacios de alta dimensionalidad. Este método consiste en evaluar combinaciones aleatorias de hiperparámetros dentro de un espacio predefinido, lo que permite cubrir una mayor diversidad de configuraciones con menos evaluaciones.

Gracias a esta simplicidad y eficiencia, Random Search permitió una exploración inicial rápida y efectiva del espacio de búsqueda, sirviendo como punto de partida para enfoques de optimización más sofisticados.

3.3.1.3 Optimización Bayesiana con TPE: Un Enfoque Adaptativo

La optimización bayesiana con TPE (Tree-structured Parzen Estimator) mejora significativamente respecto a métodos aleatorios al aprender del desempeño de configuraciones previas y ajustar su estrategia de búsqueda en función de estos resultados. Funciona en dos fases: primero explora aleatoriamente el espacio de hiperparámetros, y luego construye distribuciones probabilísticas para dirigir nuevas evaluaciones hacia las regiones más prometedoras, maximizando la mejora esperada en cada iteración.

Ventajas clave en este trabajo:

- **Eficiencia en la exploración:** TPE reduce el número de evaluaciones necesarias al enfocar la búsqueda en configuraciones con mayor probabilidad de éxito. Esto es especialmente valioso cuando se trabaja con modelos costosos de entrenar (Watanabe, 2023).
- **Manejo de interacciones complejas:** A diferencia de enfoques que tratan los hiperparámetros de forma independiente, TPE puede capturar interacciones no lineales entre ellos. Esto resulta crucial en arquitecturas profundas, donde dichas relaciones impactan directamente en el rendimiento (Watanabe, 2023).
- **Adaptabilidad a distintas escalas y distribuciones:** Gracias a su naturaleza no paramétrica, TPE puede optimizar hiperparámetros definidos en escalas logarítmicas o con distribuciones no uniformes. Aprende la estructura del espacio de búsqueda y ajusta su estrategia de muestreo de forma dinámica (Watanabe, 2023).

Esta metodología demostró ser efectiva en pruebas preliminares, logrando identificar configuraciones óptimas de forma consistente sin comprometer la viabilidad computacional. Por estas razones, se optó por adoptar TPE como método principal de optimización en este trabajo. Su capacidad para adaptarse a espacios de búsqueda complejos y costosos lo convierte en una alternativa claramente superior a la búsqueda aleatoria.

3.3.2 Espacio de Hiperparámetros Considerado por Modelo

Para entrenar cada modelo, se definió un espacio de hiperparámetros que permite optimizar su rendimiento en la tarea de forecasting. Los valores elegidos buscan un equilibrio entre capacidad de aprendizaje, estabilidad y eficiencia computacional. A continuación, se detallan los principales hiperparámetros utilizados para cada modelo y su justificación.

3.3.2.1 Simple FeedForward: Multilayer Perceptron

El modelo Simple Feedforward es una arquitectura basada en perceptrones multicapa (MLP) diseñada para capturar patrones temporales mediante el procesamiento de ventanas de datos históricos. A continuación, se detalla el rol de cada hiperparámetro en el equilibrio entre capacidad predictiva y generalización:

Arquitectura del modelo		
Hiperparámetro	Descripción Técnica	Impacto en el Modelo
hidden dimensions - categórica	Configuración de capas ocultas (neurona por capa). Valores: [20, 20], [40, 40], [100, 50].	Capas más pequeñas ([20,20]) reducen el riesgo de sobreajuste pero limitan la capacidad de modelar patrones complejos. Configuraciones profundas ([100,50]) mejoran el ajuste pero requieren mayor volumen de datos para generalizar.

Optimización del Entrenamiento		
Hiperparámetro	Descripción Técnica	Impacto en el Modelo
learning rate - flotante	Tasa de aprendizaje. Rango: [0.001, 0.01] (búsqueda en escala logarítmica).	Valores bajos (0.001) garantizan estabilidad pero convergencia lenta. Valores altos (0.01) aceleran el entrenamiento pero pueden oscilar alrededor de óptimos.
weight decay - flotante	Regularización L2. Rango: [1e-8, 1e-4] (búsqueda en escala logarítmica).	Valores bajos (1e-8) preservan la capacidad expresiva del modelo. Valores altos (1e-4) controlan el sobreajuste pero pueden subentrenar.

Configuración del Entrenamiento		
Hiperparámetro	Descripción Técnica	Impacto en el Modelo
batch normalization - categórica	Normalización por lotes. Valor fijo: True.	Estabiliza el entrenamiento acelerando la convergencia y reduciendo la sensibilidad a la inicialización de pesos.
batch size - categórica	Muestras por lote. Valores: 32, 64.	Tamaños pequeños (32) mejoran la generalización, mientras que tamaños grandes (64) optimizan el uso de hardware.
number of batches per epoch - categórica	Iteraciones por época. Valores: 50, 100.	Mayor número de lotes (100) refina los pesos pero incrementa el costo computacional por época.

Configuración Temporal		
Hiperparámetro	Descripción Técnica	Impacto en el Modelo
context length - categórica	Ventana histórica de entrada. Valores: 5×, 10×, 15× la longitud de predicción.	Ventanas cortas (5×) son eficientes para series volátiles, mientras que ventanas largas (15×) capturan estacionalidad anual a costa de mayor costo computacional.

Parámetros Fijos Clave	
Hiperparámetro	Descripción Técnica
prediction length	Horizonte de pronóstico (ej. 30 días).
maximum epochs	Límite de épocas de entrenamiento (40).

Tabla 1. Espacio de hiperparámetros considerado - Simple FeedForward

Justificación de Diseño:

El espacio de búsqueda de hiperparámetros para el modelo Simple FeedForward se diseñó para optimizar el equilibrio entre capacidad predictiva, eficiencia y generalización.

- Capacidad Predictiva: Se exploraron diversas arquitecturas de capas ocultas (desde [20, 20] hasta [100, 50]) y longitudes de contexto (hasta 15×) para permitir la captura de patrones temporales de diferente complejidad y horizonte. Configuraciones más profundas y ventanas más amplias buscan modelar series complejas y estacionales.

- Eficiencia del Entrenamiento: La tasa de aprendizaje ([0.001, 0.01] en escala logarítmica) se ajustó para una convergencia estable y eficiente. Los tamaños de lote (32 y 64) se consideraron para balancear la generalización y el uso de recursos computacionales.
- Generalización: La regularización L2 (weight decay en escala logarítmica [1e-8, 1e-4]) y la normalización por lotes (fijada en True) se incluyeron para controlar el sobreajuste y mejorar la capacidad del modelo para aplicarse a datos no vistos.

El espacio de búsqueda abarca configuraciones que buscan un modelo Simple FeedForward robusto, capaz de realizar predicciones precisas en una variedad de series temporales, optimizando tanto su rendimiento como su capacidad de generalización.

3.3.2.2 DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks

El modelo DeepAR es una arquitectura basada en redes LSTM/GRU que combina capacidades autoregresivas con aprendizaje de patrones temporales complejos. A continuación se presenta el análisis técnico de los hiperparámetros configurados:

Arquitectura del modelo		
Hiperparámetro	Descripción Técnica	Impacto en el Modelo
number of layers - categórica	Número de capas recurrentes. Valores: 1, 2, 3.	Capas adicionales (3) permiten modelar dependencias jerárquicas pero aumentan el riesgo de sobreajuste en series cortas.
hidden size - categórica	Neuronas por capa recurrente. Valores: 32, 64, 128.	Dimensiones mayores (128) capturan patrones complejos a costa de mayor costo computacional.

Optimización del Entrenamiento		
Hiperparámetro	Descripción Técnica	Impacto en el Modelo
learning rate - flotante	Tasa de aprendizaje. Rango: [0.001, 0.01] (búsqueda en escala logarítmica).	Valores bajos (0.001) aseguran estabilidad en RNNs, mientras que 0.01 puede causar divergencia en gradientes.
weight decay - flotante	Regularización L2. Rango: [1e-8, 1e-4] (búsqueda en escala logarítmica).	Controla sobreajuste sin penalizar excesivamente la capacidad recurrente del modelo.

Configuración del Entrenamiento		
Hiperparámetro	Descripción Técnica	Impacto en el Modelo
dropout rate - flotante	Dropout en capas recurrentes. Rango: [0.1, 0.5]. (búsqueda en escala normal).	Valores altos (0.5) son críticos para evitar memorización en series con alta volatilidad.
batch size - categórica	Muestras por lote. Valores: 32, 64.	Tamaños mayores (64) aceleran entrenamiento pero reducen estocasticidad beneficiosa.

Configuración Temporal		
Hiperparámetro	Descripción Técnica	Impacto en el Modelo
context length - categórica	Ventana histórica de entrada. Valores: 2×, 3×, 4×, 6× el horizonte de predicción	Ventanas más cortas (2×) ofrecen eficiencia computacional y funcionan bien en series con patrones locales o alta rotación. Ventanas más largas (4× o 6×) permiten capturar estacionalidades prolongadas o efectos de largo plazo, a costa de mayor complejidad y tiempo de entrenamiento.

Parámetros Fijos Clave	
Hiperparámetro	Descripción Técnica
number of parallel samples	100 muestras para generar intervalos probabilísticos.
time features	Características temporales personalizadas (definidas en sección de Ingeniería de atributos).
maximum epochs	Límite de 40 épocas de entrenamiento.
prediction length	Horizonte de pronóstico (ej: 30 días).

Tabla 2. Espacio de hiperparámetros considerado - DeepAR

Justificación de Diseño:

El espacio de búsqueda de hiperparámetros para el modelo DeepAR se diseñó para equilibrar la capacidad predictiva, la eficiencia del entrenamiento y la generalización en series temporales.

- Capacidad Predictiva: Se exploraron diferentes profundidades (1-3 capas) y tamaños de capas recurrentes (32-128 neuronas) para permitir la captura de dependencias temporales complejas y jerárquicas. Ventanas de contexto variables (hasta $6 \times$ el horizonte de predicción) se consideraron para modelar patrones tanto a corto como a largo plazo, incluyendo posibles estacionalidades.
- Eficiencia del Entrenamiento: La tasa de aprendizaje ([0.001, 0.01] en escala logarítmica) se ajustó para asegurar una convergencia estable en la arquitectura RNN. Los tamaños de lote (32 y 64) se evaluaron para optimizar el uso de recursos sin sacrificar la estocasticidad beneficiosa para la generalización.
- Generalización: Se incluyó la regularización L2 (weight decay en escala logarítmica [1e-8, 1e-4]) para controlar el sobreajuste inherente a las RNNs. El dropout en las capas recurrentes (rango [0.1, 0.5]) se consideró crucial para prevenir la memorización excesiva, especialmente en series con alta variabilidad.

El espacio de búsqueda abarca configuraciones que buscan un modelo DeepAR robusto, capaz de realizar pronósticos probabilísticos precisos en diversas series temporales, gestionando tanto la complejidad de los patrones como el riesgo de sobreajuste.

3.3.2.3 Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting

Este modelo combina mecanismos de atención y componentes recurrentes para generar predicciones interpretables. A continuación se presenta el análisis técnico de los hiperparámetros configurados:

Arquitectura del modelo		
Hiperparámetro	Descripción Técnica	Impacto en el Modelo
hidden dimensions - categórica	Dimensión de capas internas. Valores: 32, 64, 128.	Mayores dimensiones (128) capturan patrones complejos pero aumentan el riesgo de sobreajuste en series pequeñas.
variable dimensions - categórica	Dimensión de embeddings. Valores: 16, 24, 32.	Afecta cómo se proyectan variables estáticas/dinámicas. 24 suele ser óptimo para datos de retail.
number of heads - categórica	Cabezas de atención. Valores: 4, 8.	Más cabezas (8) mejoran la captura de relaciones multivariadas, pero con

		costo computacional cuadrático.
--	--	---------------------------------

Optimización del Entrenamiento		
Hiperparámetro	Descripción Técnica	Impacto en el Modelo
learning rate - flotante	Tasa de aprendizaje. Rango: [0.001, 0.01] (búsqueda en escala logarítmica).	Valores altos (0.01) pueden causar inestabilidad en mecanismos de atención.
weight decay - flotante	Regularización L2. Rango: [1e-8, 1e-4] (búsqueda en escala logarítmica).	Controla sobreajuste sin penalizar excesivamente las conexiones de atención.

Configuración del Entrenamiento		
Hiperparámetro	Descripción Técnica	Impacto en el Modelo
dropout rate - flotante	Dropout en capas. Rango: [0.15, 0.35] (búsqueda en escala normal).	Valores altos (0.35) son críticos para evitar sobreajuste en datos ruidosos.
batch size - categórica	Muestras por lote. Valores: 32, 64.	64 acelera entrenamiento pero reduce estocasticidad beneficiosa.

Configuración Temporal		
Hiperparámetro	Descripción Técnica	Impacto en el Modelo
context length - categórica	Ventana histórica de entrada. Valores: 2×, 3×, 4×, 6× el horizonte de predicción.	Ventanas más largas (hasta 6×) permiten capturar patrones estacionales complejos, pero aumentan el uso de memoria GPU y el tiempo de entrenamiento.

Parámetros Fijos Clave	
Hiperparámetro	Descripción Técnica
maximum epochs	Límite de 40 épocas de entrenamiento.
time features	Características temporales personalizadas (definidas en sección de Ingeniería de atributos).

prediction length	Horizonte de pronóstico (ej: 30 días).
--------------------------	--

Tabla 3. Espacio de hiperparámetros considerado - Temporal Fusion Transformers

Justificación de Diseño:

El espacio de búsqueda de hiperparámetros para el modelo Temporal Fusion Transformer (TFT) se diseñó para equilibrar la capacidad predictiva, la eficiencia del entrenamiento y la generalización, considerando su arquitectura basada en atención.

- **Capacidad Predictiva:** Se exploraron diferentes dimensiones internas (hidden dimensions de 32 a 128) y número de cabezas de atención (4 y 8) para permitir la captura de relaciones complejas entre variables y patrones temporales. Ventanas de contexto variables (hasta $6 \times$ el horizonte de predicción) se consideraron para modelar dependencias a largo plazo y estacionalidades.
- **Eficiencia del Entrenamiento:** La tasa de aprendizaje ([0.001, 0.01] en escala logarítmica) se ajustó para asegurar la estabilidad del entrenamiento con mecanismos de atención. Los tamaños de lote (32 y 64) se evaluaron para optimizar el uso de memoria y la velocidad de convergencia.
- **Generalización:** Se incluyó una tasa de dropout ([0.15, 0.35]) para prevenir el sobreajuste, especialmente en datos potencialmente ruidosos. La regularización L2 (weight decay en escala logarítmica [1e-8, 1e-4]) se aplicó para controlar la complejidad del modelo sin penalizar excesivamente las capacidades de atención.

El espacio de búsqueda para el TFT busca identificar una configuración que explote su capacidad para modelar relaciones multivariadas y realizar pronósticos interpretables, manteniendo un equilibrio entre el rendimiento predictivo y la robustez ante datos no vistos.

3.3.2.4 WaveNet: A Generative Model for Raw Audio

Aunque originalmente diseñado para el procesamiento de audio, WaveNet ha demostrado ser eficaz en forecasting de series temporales gracias a su arquitectura de convoluciones dilatadas. A continuación se presenta el análisis técnico de los hiperparámetros configurados:

Arquitectura del modelo		
Hiperparámetro	Descripción Técnica	Impacto en el Modelo
number of residual channels - categórica	Canales en capas residuales. Valores: 24, 32.	Mayores canales (32) capturan patrones complejos pero incrementan memoria GPU.
number of skip channels - categórica	Canales en capas de salida. Valores: 32, 48.	Valores altos (48) mejoran la reconstrucción de patrones no lineales.
number of stacks - categórica	Bloques de convolución apilados. Valores: 1, 2	Más bloques permiten captar dependencias largas,

		pero aumentan el tiempo de entrenamiento.
number of bins - categórica	Cuantización de valores. Valores: 512, 1024.	Más bins representan mejor series volátiles; menos bins reducen memoria.
embedding dimension - categórica	Dimensión de embeddings. Valores: 5, 10.	Dimensiones altas capturan relaciones complejas; bajas reducen el costo.

Optimización del Entrenamiento		
Hiperparámetro	Descripción Técnica	Impacto en el Modelo
learning rate - flotante	Tasa de aprendizaje. Rango: [0.001, 0.01] (búsqueda en escala logarítmica).	Valores altos pueden generar inestabilidad en entrenamiento.
weight decay - flotante	Regularización L2. Rango: [1e-8, 1e-4] (búsqueda en escala logarítmica).	Ayuda a evitar sobreajuste sin limitar la expresividad.

Configuración del Entrenamiento		
Hiperparámetro	Descripción Técnica	Impacto en el Modelo
batch size - categórica	Muestras por lote. Valores: 32, 64.	Lotes pequeños mejoran estabilidad; grandes aceleran el entrenamiento.
number of batches per epoch - categórica	Iteraciones por época. Valores: 50, 100.	Más iteraciones ajustan mejor el modelo pero requieren más tiempo.

Parámetros Fijos Clave	
Hiperparámetro	Descripción Técnica
gradient clip value	Limita gradientes a 0.1 (crítico para convoluciones dilatadas).
maximum epochs	Límite de 40 épocas de entrenamiento.
time features	Características temporales personalizadas (definidas en sección de Ingeniería de atributos).
prediction length	Horizonte de pronóstico (ej: 30 días).

use logarithmic scale feature	True. Transformación logarítmica para estabilizar magnitudes.
frequency	Frecuencia de los datos (diaria/semanal). Define cómo se interpretan las marcas temporales.

Tabla 4. Espacio de hiperparámetros considerado - WaveNet

Justificación de Diseño:

El espacio de búsqueda de hiperparámetros para el modelo WaveNet se diseñó para equilibrar la capacidad predictiva, la eficiencia del entrenamiento y la generalización, considerando su arquitectura basada en convoluciones dilatadas.

- Capacidad Predictiva: Se exploraron diferentes profundidades (1-2 stacks) y capacidades de modelado (number of residual channels de 24-32, number of skip channels de 32-48, number of bins de 512-1024) para capturar patrones complejos y la volatilidad inherente en las series temporales. Una mayor capacidad se consideró para modelar con precisión series no lineales.
- Eficiencia del Entrenamiento: La tasa de aprendizaje ([0.001, 0.01] en escala logarítmica) se ajustó para promover la estabilidad del entrenamiento en una arquitectura profunda. El tamaño del lote (32 y 64) y el número de lotes por época (50 y 100) se consideraron para optimizar la velocidad de convergencia sin comprometer la estabilidad, reforzada por el recorte de gradientes.
- Generalización: Se incluyó la regularización L2 (weight decay en escala logarítmica [1e-8, 1e-4]) para prevenir el sobreajuste. La transformación logarítmica de la característica (use logarithmic scale feature=True) se fijó para estabilizar las magnitudes variables de las series, facilitando un mejor aprendizaje y generalización.

El espacio de búsqueda para WaveNet busca identificar una configuración que explote su capacidad para modelar series temporales complejas e irregulares, manteniendo un equilibrio entre la precisión predictiva y la robustez ante la variabilidad de los datos.

3.3.3 Control de Sobreajuste de Entrenamiento

Para prevenir el sobreajuste —es decir, que el modelo aprenda patrones esporádicos del conjunto de entrenamiento y pierda capacidad de generalización— se implementó la técnica de Early Stopping durante el entrenamiento. Todos los modelos fueron configurados con un máximo de 40 épocas y un criterio de paciencia de 10 épocas. Esto significa que, si la pérdida (loss) sobre el conjunto de validación no mejoraba durante diez épocas consecutivas, el entrenamiento se interrumpía de forma automática.

Durante la búsqueda de hiperparámetros mediante Optimización Bayesiana, se monitoreó la pérdida de validación en cada iteración y se aplicó Early Stopping bajo estos mismos criterios. Además de seleccionar la mejor configuración de hiperparámetros, se

registró cuántas épocas fueron necesarias para alcanzar la métrica mínima de validación, así como el número total de épocas ejecutadas hasta la detención definitiva.

Una vez identificada la configuración óptima, los modelos fueron reentrenados utilizando la totalidad de los datos disponibles para entrenamiento, combinando los subconjuntos de entrenamiento y validación. En esta fase, se respetó la misma cantidad de épocas observada durante la búsqueda, incluyendo aquellas adicionales correspondientes al parámetro de paciencia.

Esta decisión se fundamentó en observaciones empíricas: si bien la mejor pérdida en validación se alcanzaba antes del corte, las épocas adicionales no mostraban señales claras de sobreajuste, manteniéndose estables en su desempeño. Por lo tanto, se consideró que mantener ese margen de entrenamiento podía beneficiar la generalización del modelo al aprovechar el mayor volumen de datos disponible, sin comprometer su robustez. Esta estrategia contribuyó a mejorar el rendimiento final en el conjunto de prueba, manteniendo la consistencia metodológica en todo el proceso.

3.4 Ingeniería de Atributos

Para enriquecer el análisis temporal y mejorar la capacidad predictiva del modelo, se generaron nuevos atributos basados en la columna "fecha_comercial". Estos atributos adicionales permiten capturar patrones temporales comunes y relevantes, los cuales son aplicables de manera uniforme a todas las tiendas. De esta forma, se garantiza que el modelo, entrenado por producto para todas las tiendas, aproveche de manera óptima las relaciones temporales dentro de los datos. A continuación, se describen los atributos generados:

- Mes (month): Extrae el número del mes (1 a 12) de la fecha.
- Día del Mes (day): Representa el día del mes (1 a 31).
- Día de la Semana (day_of_week): Indica el día de la semana como número (0 para lunes, 6 para domingo).
- Semana del Año (week_of_year): Extrae el número de la semana del año correspondiente a la fecha.
- Día del Año (day_of_year): Indica el número del día dentro del año (1 a 365/366).

3.5 Evaluación de Modelos

3.5.1 Métrica de Evaluación

El objetivo del modelo es predecir la cantidad de unidades vendidas para cada combinación de Fecha Comercial, Tienda y Producto. Para ello, se definió una ventana de predicción de 30 días, dentro de la cual se evaluó el desempeño de los distintos modelos.

Para comparar los modelos y seleccionar el más adecuado, se utilizó el Error Cuadrático Medio Raíz (RMSE) como métrica principal de evaluación. El RMSE se define como:

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

Donde:

- y_i representa el valor real de ventas en el día i
- \hat{y}_i representa la predicción del modelo para ese mismo día.
- n es el número total de predicciones evaluadas.

El RMSE presenta varias ventajas que justifican su elección como métrica de evaluación:

- Interpretabilidad directa: Expresa el error en las mismas unidades que la variable objetivo, facilitando la interpretación de los resultados.
- Penalización de errores grandes: Al elevar las diferencias al cuadrado antes de promediarlas, el RMSE otorga mayor peso a los errores grandes, lo que es clave en forecasting, ya que predicciones con desvíos significativos pueden afectar la toma de decisiones comerciales.
- Amplio uso en series temporales: Es una métrica estándar en problemas de forecasting, lo que permite comparar el rendimiento con estudios previos y metodologías similares.

Dado que el objetivo es minimizar la diferencia entre las predicciones y los valores reales en la ventana de 30 días, el RMSE se calcula para cada combinación de Producto-Tienda.

3.5.2 Metodología de Evaluación

Dado que los modelos fueron entrenados a nivel de producto —es decir, un modelo por cada producto— y considerando la distribución de productos en los distintos clústeres (100 productos en el Clúster 0, 13 en el Clúster 1 y 41 en el Clúster 2), se identificaron los 10 productos con mayor participación en las tiendas por cada clúster para evaluar el desempeño de los modelos. Los productos seleccionados de cada clúster fueron los siguientes:

Cluster 0	
Nombre Producto	Participación Tiendas
REFRIG COCA COLA LT 350ML	30
REFRIG GUARANA ANTARCTICA LT 350ML	28
REFRIG COCA COLA SA LT 350ML	26
REFRIG COCA COLA LT 220ML	22

REFRIG FANTA LARANJA LT 350ML	20
REFRIG COCA COLA SA PET 1,5L	19
REFRIG GUARANA ANTARCTICA ZERO LT 350ML	19
REFRIG PEPSI COLA LT 350ML	19
REFRIG COCA COLA CAFE EXPRESSO LT 220ML	18
REFRIG PEPSI COLA TWIST PET 2L	18

Total Series A Predecir	219
--------------------------------	------------

Cluster 1	
Nombre Producto	Participación Tiendas
REFRIG COCA COLA PET 2,5L	26
REFRIG GUARANA ANTARCTICA PET 2L	26
CERVEJA ANTARCTICA PILSEN LATA 350ML	25
CERVEJA CRYSTAL PILSEN LATA 350ML	25
CERVEJA SKOL PILSEN LATA 350ML	25
REFRIG FANTA LARANJA PET 2,25L	20
REFRIG COCA COLA SA PET 2,25L	19
CERVEJA BRAHMA CHOPP LATA CX18X 350ML	15
REFRIG POTY GUARANA PET 2L	9
REFRIG COCA COLA SA PET 2L	4

Total Series A Predecir	198
--------------------------------	------------

Cluster 2	
Nombre Producto	Participación Tiendas
CERVEJA BRAHMA CHOPP LATA 350ML	27
CERVEJA HEINEKEN LONG NECK 330ML	27
REFRIG COCA COLA PET 600ML	26
CERVEJA PETRA PURO MALTE LATA 350ML	25
CERVEJA BUDWEISER LATA 350ML	22
REFRIG FANTA UVA PET 2L	20
REFRIG SPRITE LIMAO PET 2L	20
REFRIG SUKITA LARANJA PET 2L	20
CERVEJA SPATEN LATA 350ML	18
REFRIG PEPSI BLACK COLA PET 2L	18
Total Series A Predecir	223

Tabla 5. Productos considerados para el análisis según cluster

De esta manera, se hicieron predicciones sobre 640 series utilizando solo 30 modelos.

Para evaluar el rendimiento de cada modelo en cada clúster, se llevó a cabo el siguiente procedimiento:

- 1) **Comparación con la Metodología Benchmark:** Cada modelo fue comparado con la metodología de venta media, considerada como referencia base. Para ello, se calculó el RMSE para cada combinación Producto-Tienda dentro de cada clúster, y se analizó lo siguiente:
 - Se calculó el porcentaje de mejora del modelo en relación con el benchmark mediante la siguiente fórmula:

$$\text{Mejora \%} = 100 \times \frac{\text{RMSE}_{\text{baseline}} - \text{RMSE}_{\text{modelo}}}{\text{RMSE}_{\text{baseline}}}$$

A partir de esta métrica, se realizó un análisis estadístico para evaluar el desempeño del modelo en comparación con el benchmark. Este análisis incluyó:

- Cálculo de estadísticas descriptivas, como la proporción de casos en los que el modelo supera al benchmark y la media y la mediana de mejora en esos casos.
 - Visualización de la distribución de la mejora mediante histogramas, permitiendo observar la variabilidad de los resultados.
 - Análisis por puntos de venta y productos a través de mapas de calor, que facilitarán la identificación de patrones de mejora en distintos escenarios.
- 2) **Comparación entre Todos los Modelos:** En una segunda instancia, se incluyeron todos los modelos junto con el benchmark, permitiendo que compitieran entre sí. El objetivo fue identificar el modelo con el RMSE más bajo en la mayor cantidad de combinaciones Producto-Tienda, determinándolo como la mejor alternativa global del clúster. Además, se incorporó un ranking basado en posiciones relativas para comparar los modelos sin depender de los valores absolutos del RMSE. En cada combinación Producto-Tienda, se ordenaron los modelos según su RMSE, asignando un ranking a cada uno, donde el mejor modelo recibió el ranking más bajo. Luego, se calculó la media de los rankings por modelo, seleccionando aquel con el ranking promedio más bajo como el modelo óptimo.
- 3) **Registro de Tiempos de Entrenamiento:** Como complemento al análisis de precisión, se registraron también los tiempos de entrenamiento necesarios para completar los modelos de cada clúster (es decir, el tiempo requerido para entrenar los 10 modelos correspondientes a los productos del clúster). Esta métrica permitió evaluar la eficiencia computacional de cada enfoque, ofreciendo una visión más completa del compromiso entre precisión y costo computacional.
- 4) **Comparación con XGBoost:** Finalmente, se comparó el modelo de mejor desempeño con XGBoost, uno de los algoritmos de machine learning más populares y ampliamente utilizados en la industria para tareas de predicción de demanda. Esta comparación se realizó bajo las mismas condiciones de entrenamiento y evaluación: XGBoost fue entrenado siguiendo el mismo enfoque de un modelo por producto, utilizando los mismos atributos de entrada, aplicando la misma metodología de optimización bayesiana de hiperparámetros y prediciendo de manera conjunta la demanda de todas las tiendas asociadas a cada producto. Esta instancia permitió analizar objetivamente el rendimiento de los modelos de deep learning desarrollados frente a una solución industrialmente consolidada, validando la calidad de los resultados obtenidos.
- Cabe mencionar que también se evaluó una variante en la que se entrenó un modelo XGBoost por cada combinación producto-tienda. Sin embargo, dado que la estrategia basada en un modelo por producto mostró un desempeño superior, se optó por mantener esta última como enfoque principal de comparación.

Este enfoque buscó garantizar una evaluación exhaustiva y objetiva del desempeño de cada modelo, permitiendo seleccionar la mejor alternativa basada en evidencia cuantitativa, eficiencia y pruebas estadísticas rigurosas.

4. Resultados Obtenidos

4.1. Comparación Modelos vs Benchmark

A continuación, se presenta la distribución del desempeño de los modelos analizados en comparación con la metodología Benchmark según cluster, incluyendo la proporción de combinaciones producto-tienda en las que obtuvo el menor RMSE, el análisis del porcentaje de mejora mediante un histograma, y un mapa de calor que desglosa dicha mejora por punto de venta y producto para visualizar su desempeño relativo.

4.1.1. Cluster 0

- Simple FeedForward vs Benchmark:

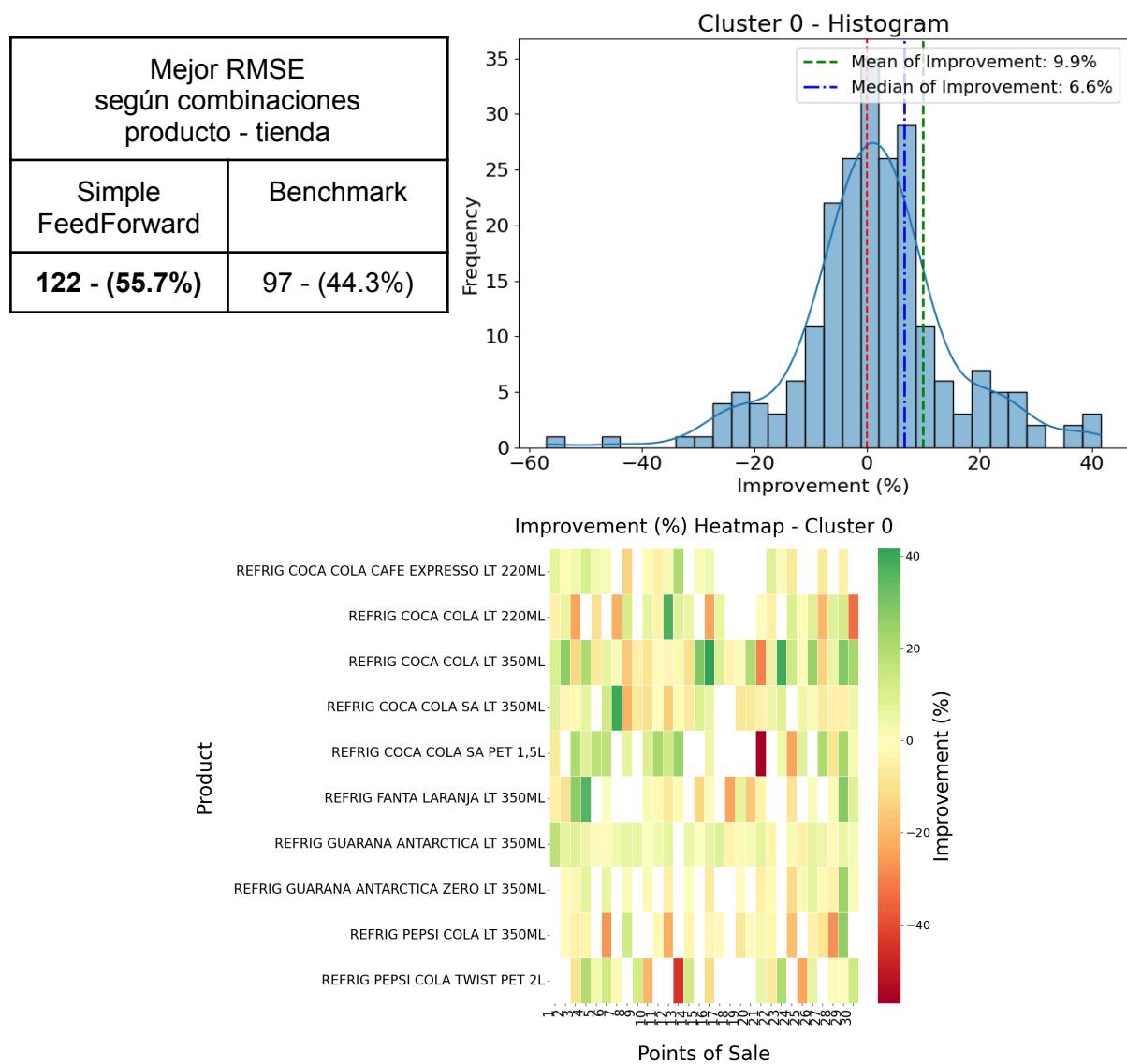


Figura 19. Desempeño de Simple FeedForward vs Benchmark - Cluster 0

- DeepAR vs Benchmark:

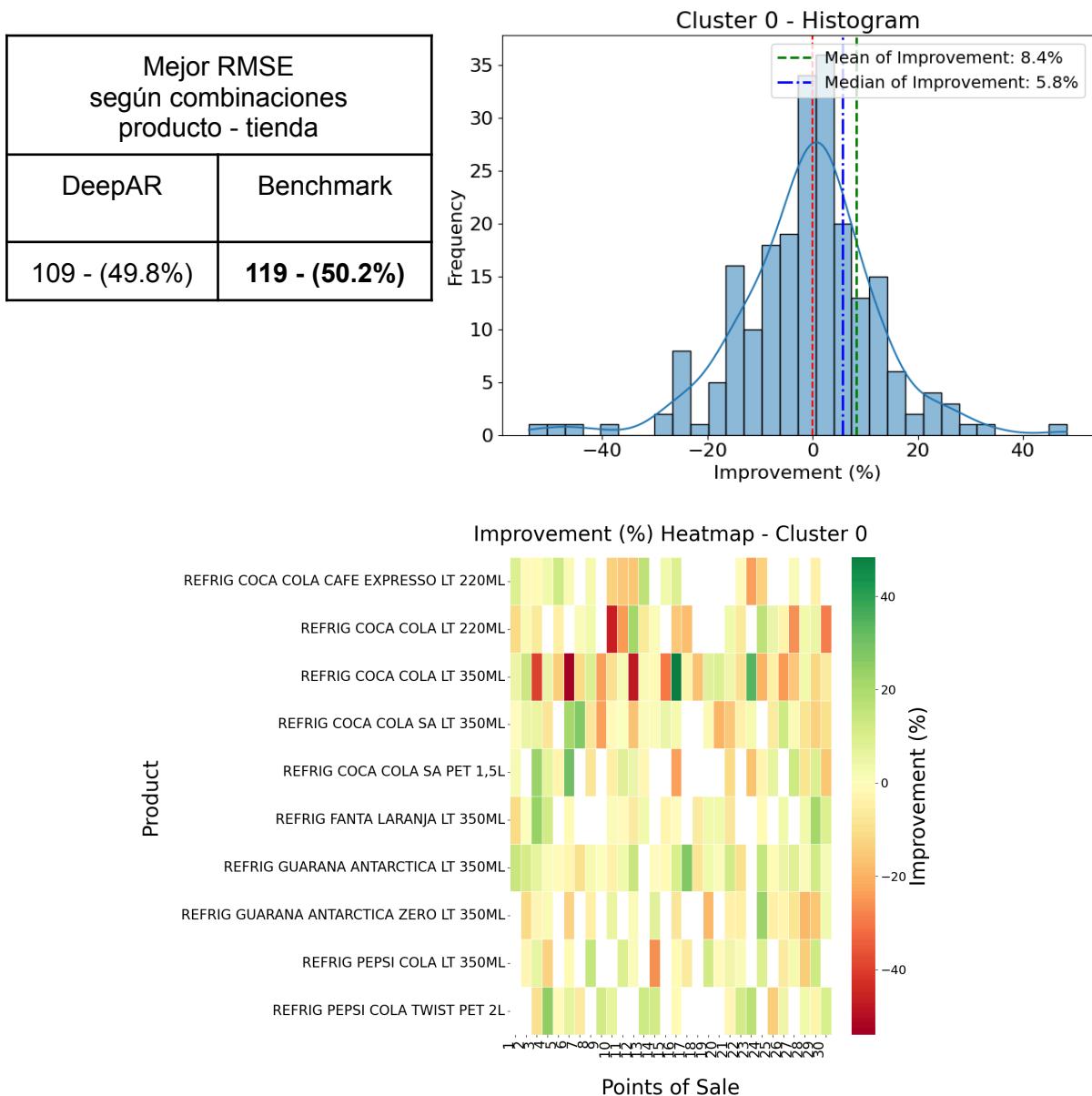


Figura 20. Desempeño de DeepAR vs Benchmark - Cluster 0

- **Temporal Fusion Transformers vs Benchmark:**

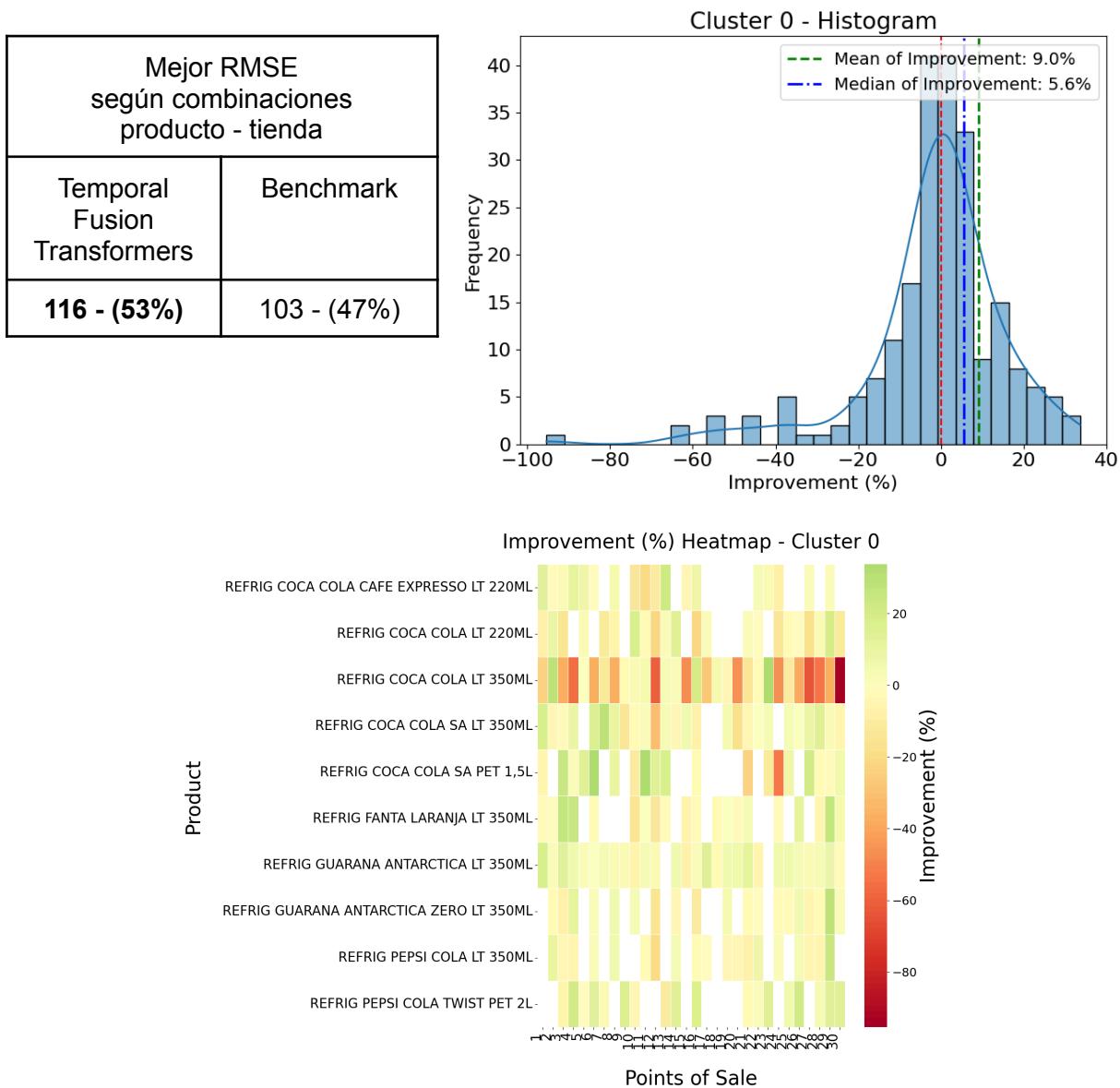


Figura 21. Desempeño de Temporal Funsion Transformers vs Benchmark - Cluster 0

- **WaveNet vs Benchmark:**

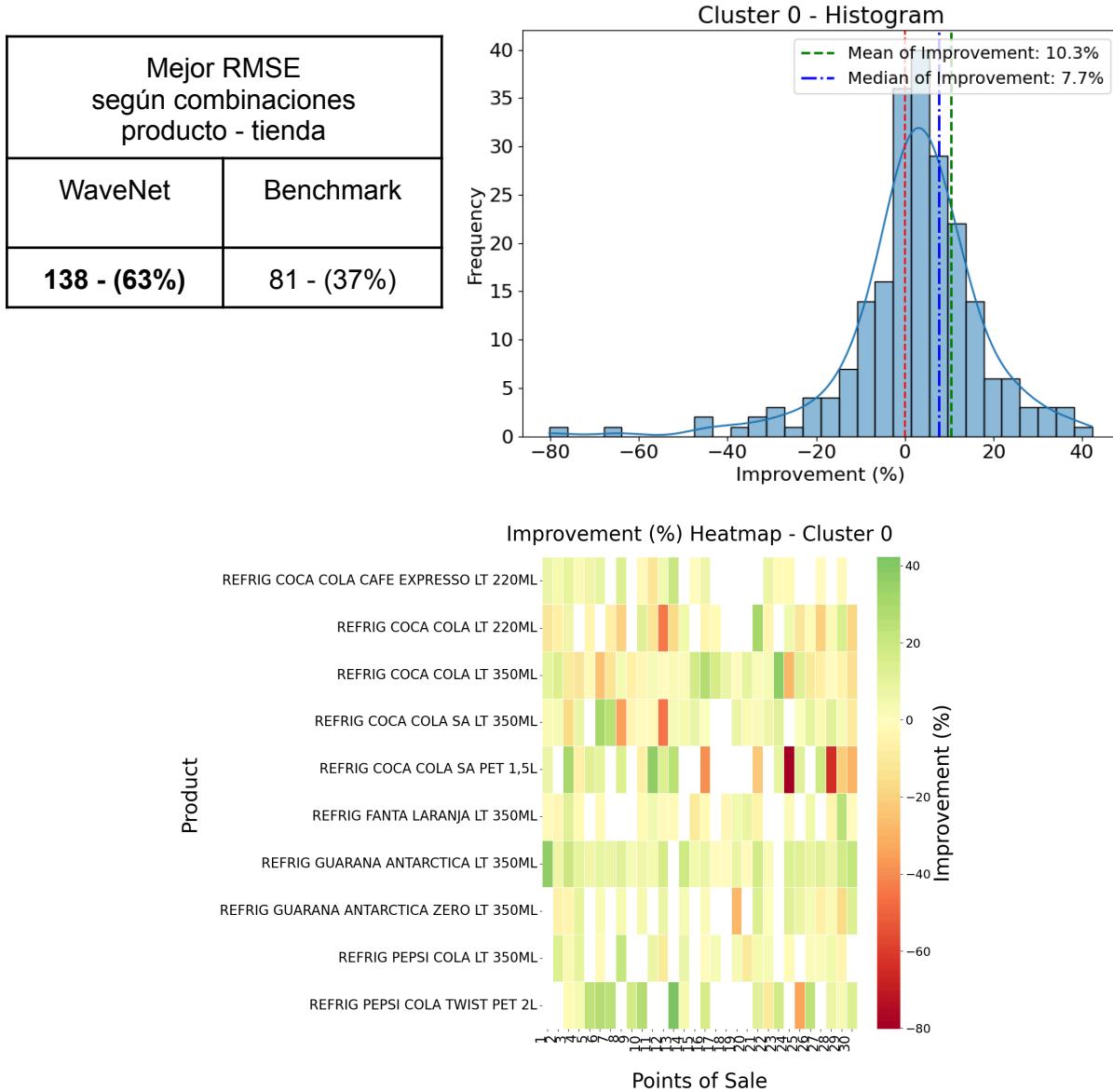


Figura 22. Desempeño de WaveNet vs Benchmark - Cluster 0

4.1.2. Cluster 1

- Simple FeedForward vs Benchmark:

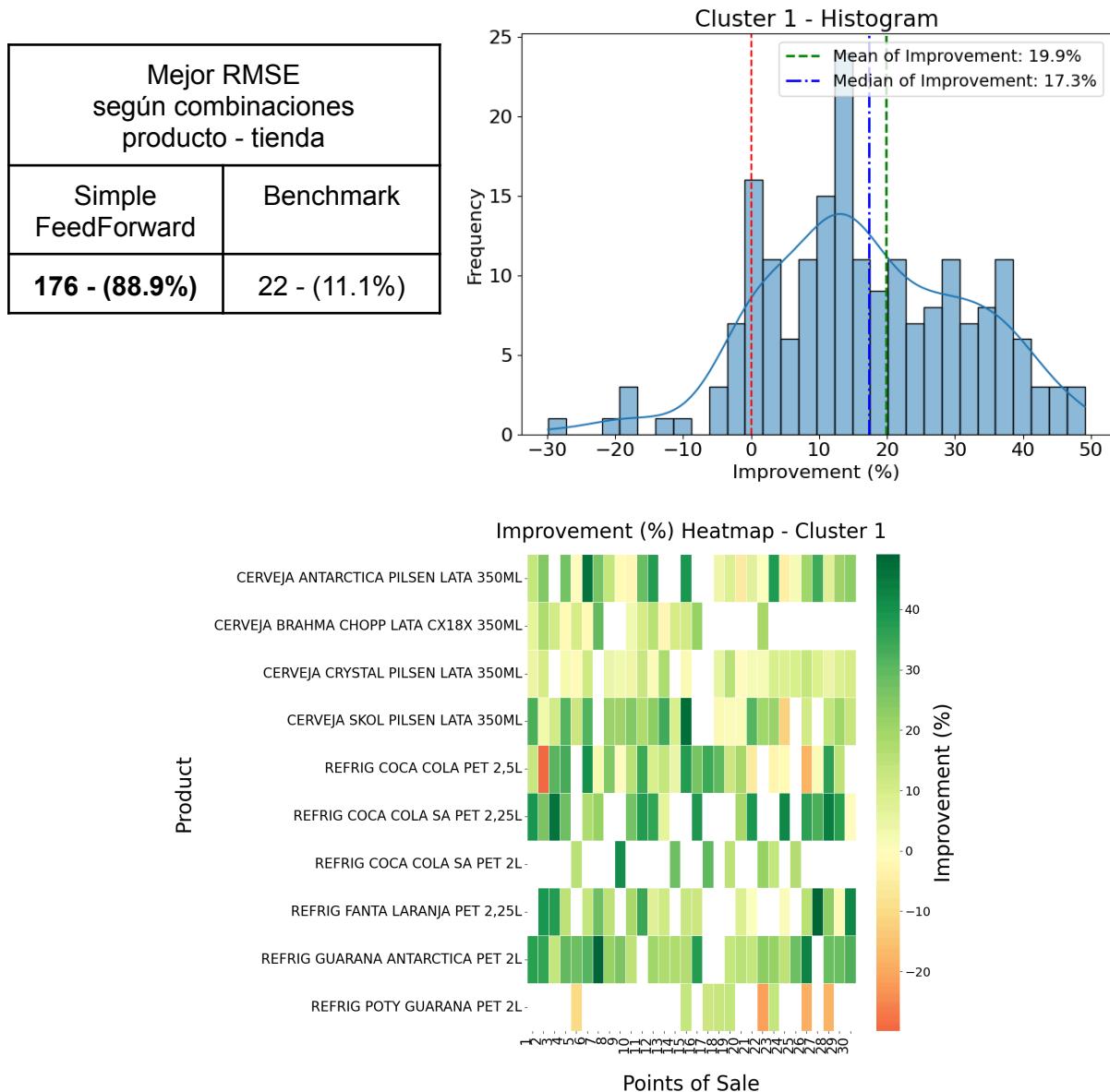


Figura 23. Desempeño de Simple FeedForward vs Benchmark - Cluster 1

- DeepAR vs Benchmark:

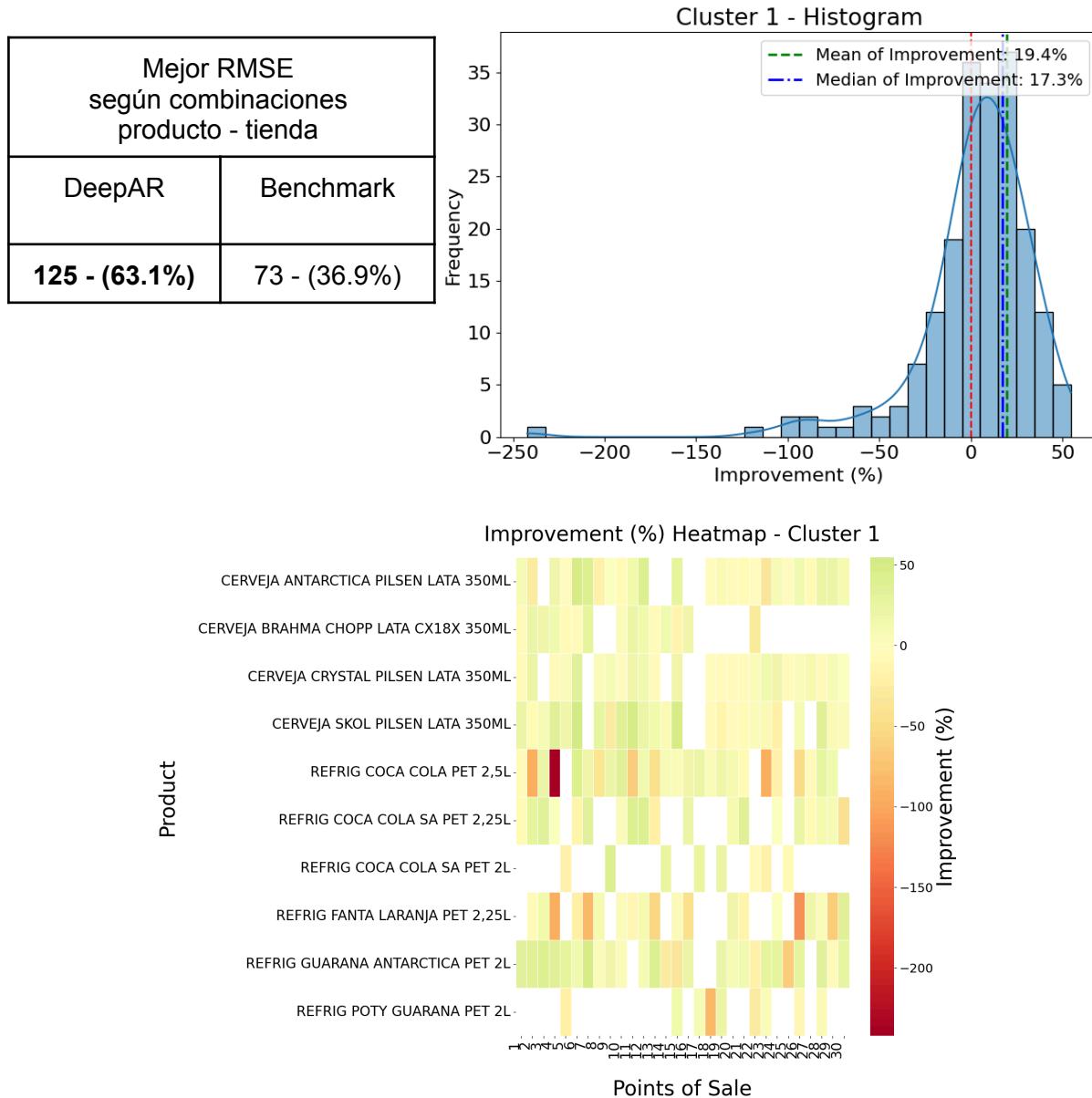


Figura 24. Desempeño de DeepAR vs Benchmark - Cluster 1

- **Temporal Fusion Transformers vs Benchmark:**

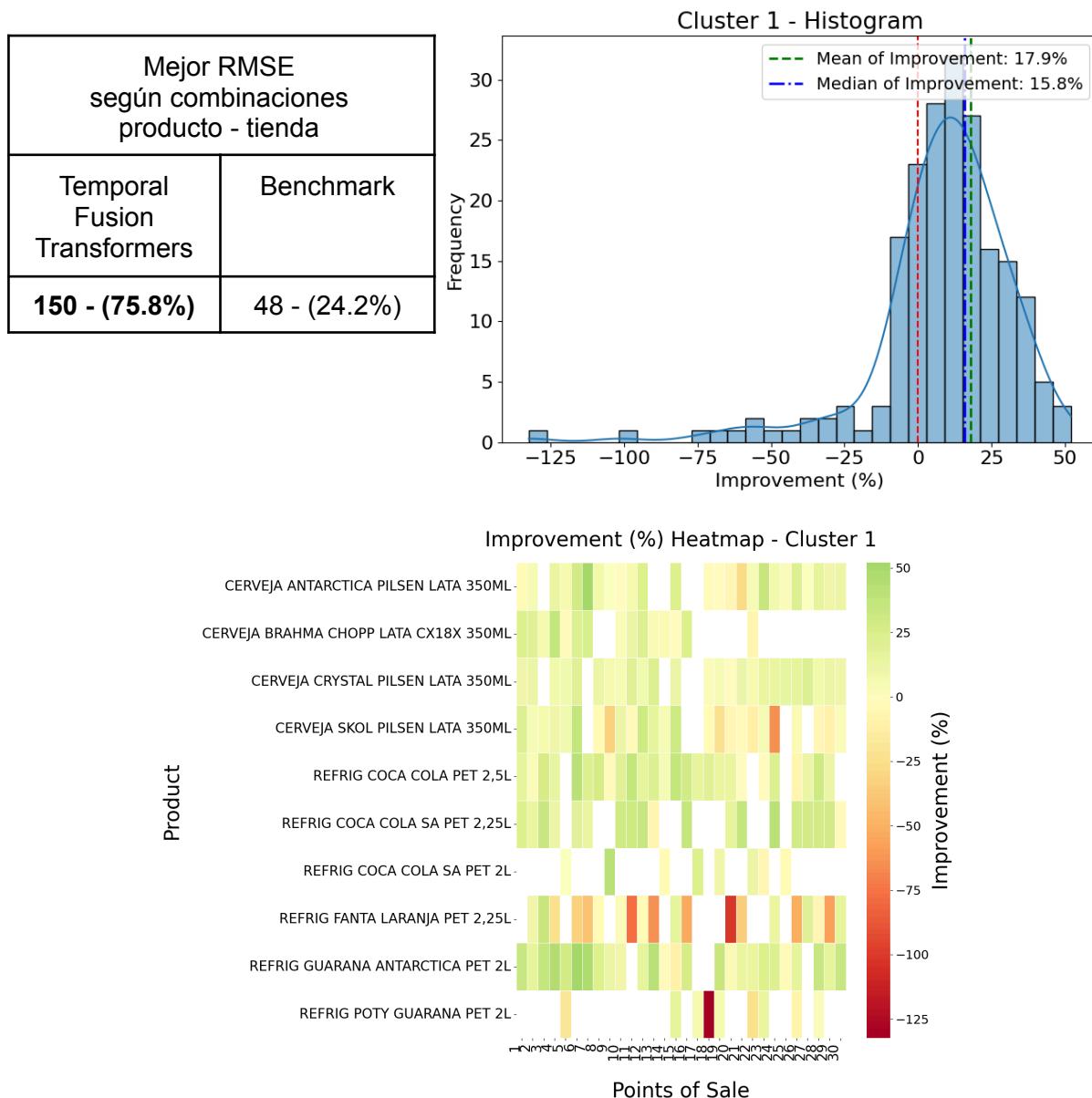


Figura 25. Desempeño de Temporal Fusion Transformers vs Benchmark - Cluster 1

- **WaveNet vs Benchmark:**

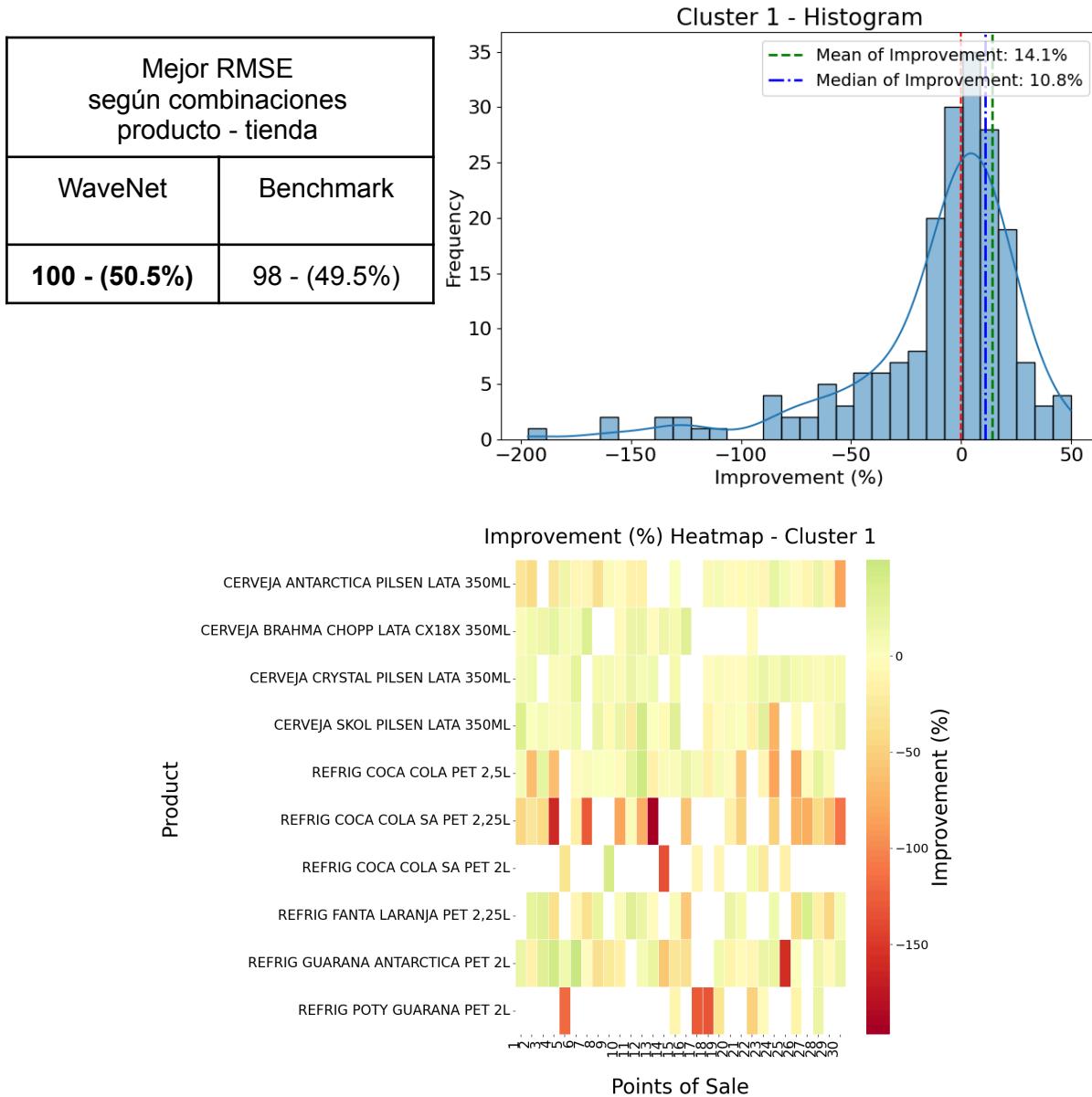


Figura 26. Desempeño de WaveNet vs Benchmark - Cluster 1

4.1.3. Cluster 2

- Simple FeedForward vs Benchmark:

Mejor RMSE según combinaciones producto - tienda	
Simple FeedForward	Benchmark
190 - (85.2%)	33 - (14.8%)

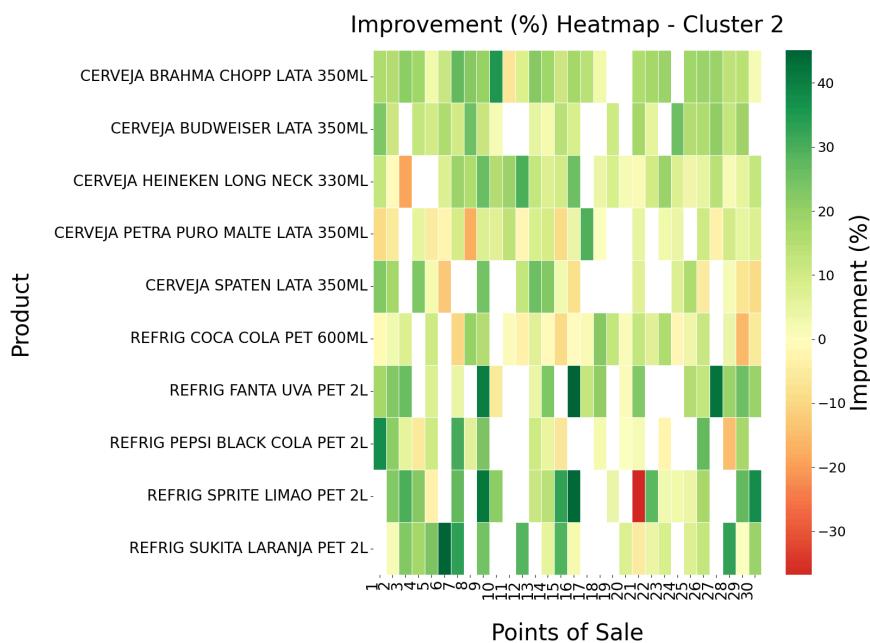
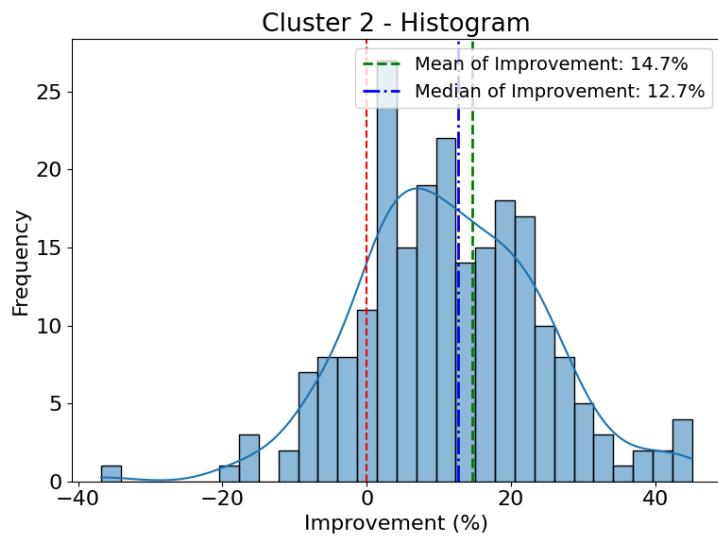


Figura 27. Desempeño de Simple FeedForward vs Benchmark - Cluster 2

- DeepAR vs Benchmark:

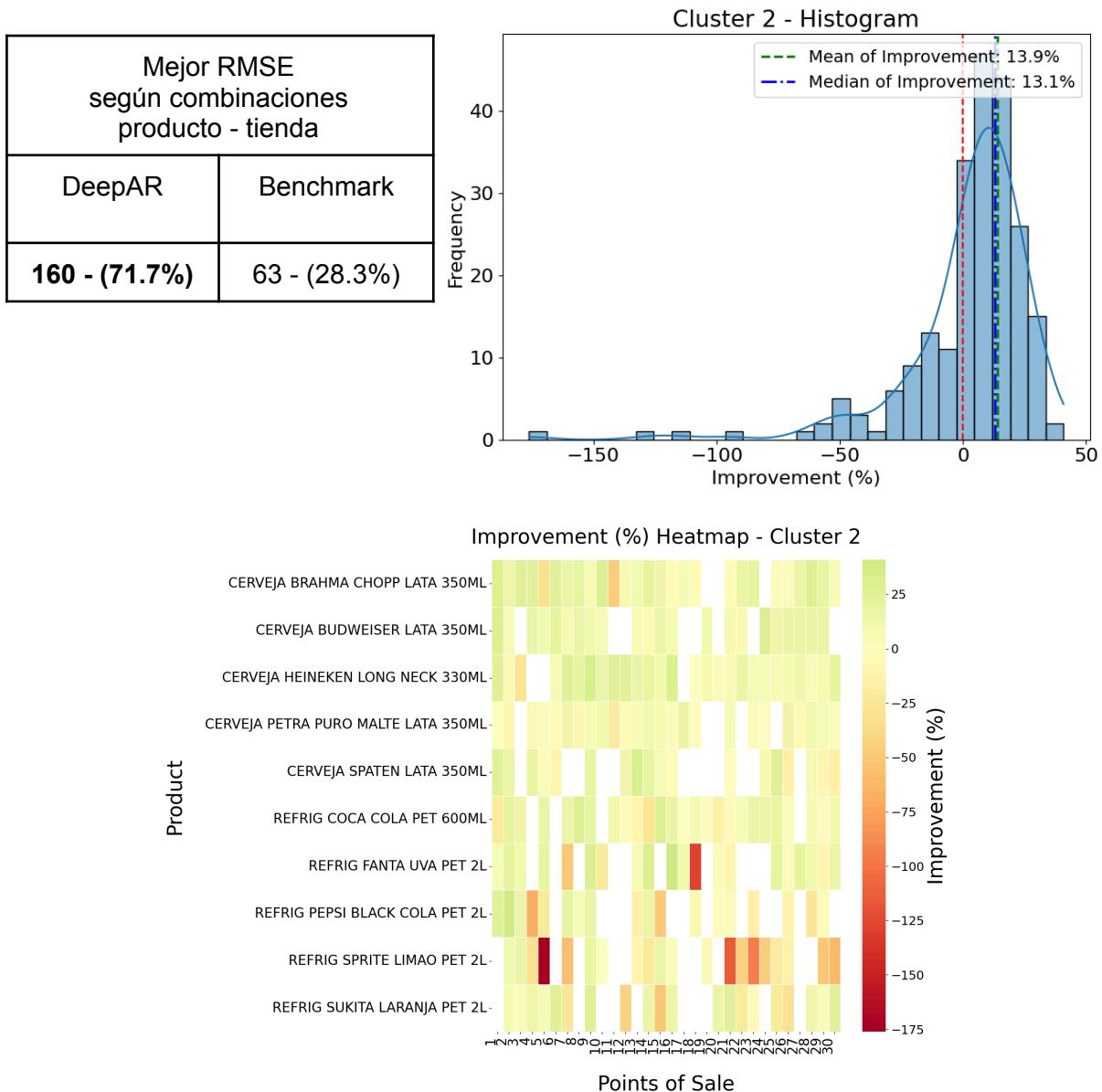


Figura 28. Desempeño de DeepAR vs Benchmark - Cluster 2

- **Temporal Fusion Transformers vs Benchmark:**

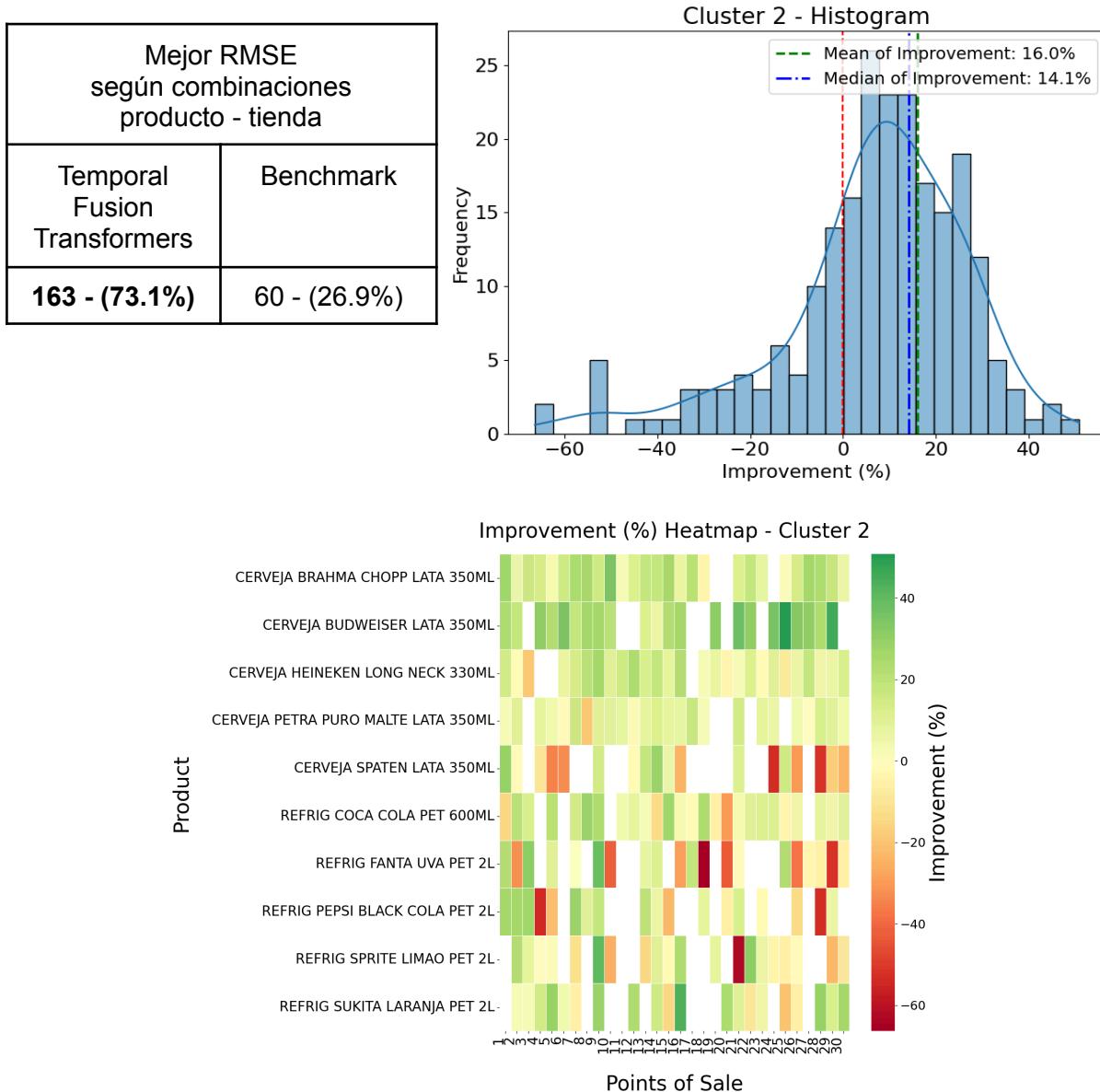


Figura 29. Desempeño de Temporal Fusion Transformers vs Benchmark - Cluster 2

- **WaveNet vs Benchmark:**

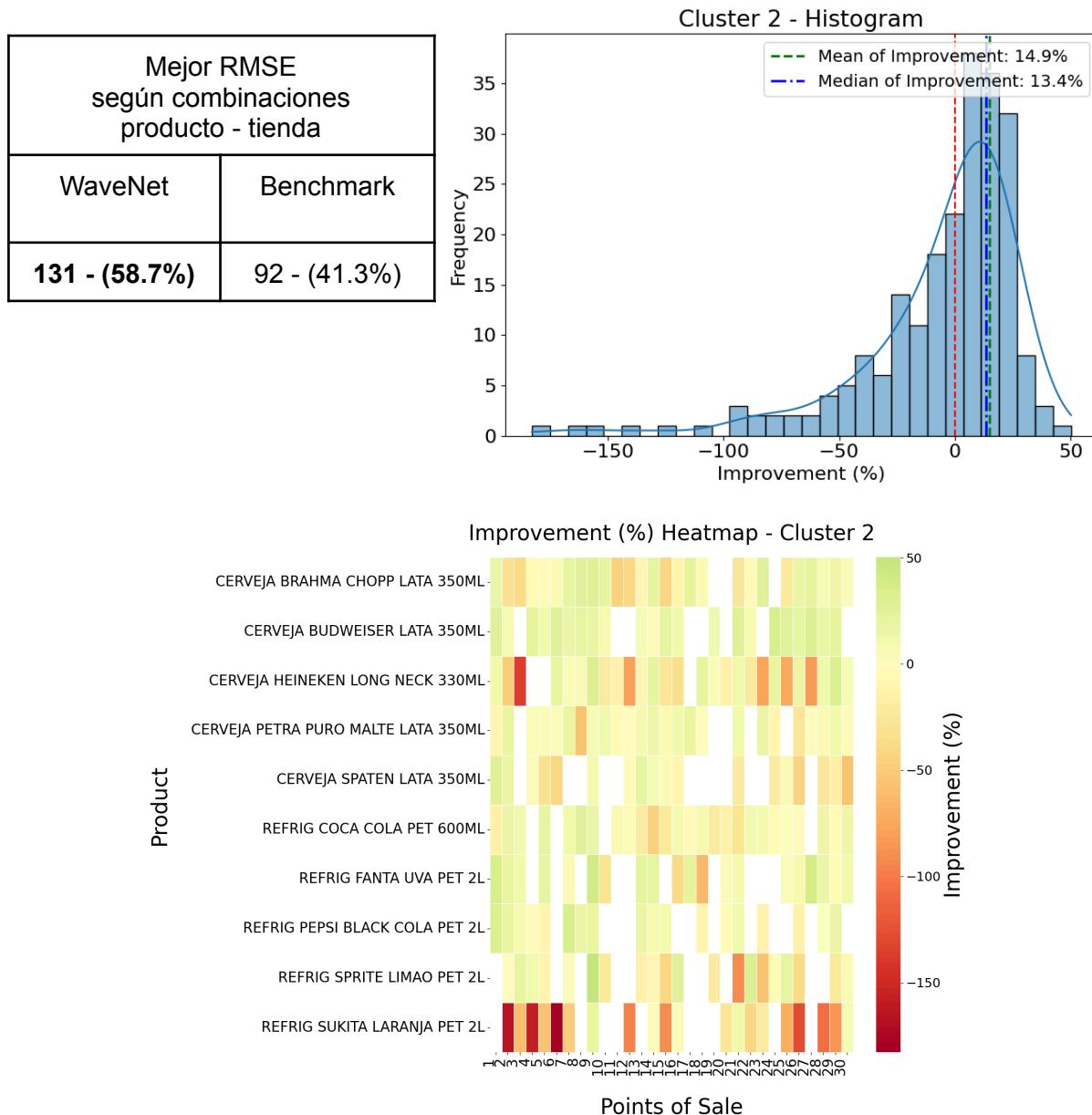


Figura 30. Desempeño de WaveNet vs Benchmark - Cluster 2

4.2. Comparación entre Todos los Modelos

A continuación, se presentan los resultados de la comparación del desempeño de todos los modelos evaluados en conjunto dentro de cada clúster. El análisis se centra en dos aspectos principales: la mediana del RMSE global alcanzado por cada modelo en cada clúster, y el ranking medio de cada modelo a través de todas las combinaciones de Producto-Tienda dentro de cada clúster, junto con la frecuencia con la que cada modelo obtuvo el RMSE más bajo.

- Análisis de mediana del RMSE global:

	Cluster 0 Mediana MRSE	Cluster 1 Mediana MRSE	Cluster 2 Mediana MRSE
SimpleFeed Forward	6287.0	30709.5	15902.0
Temporal Fusion Transformer	6552.0	33370.1	17798.1
DeepAR	6574.2	34277.7	18070.3
WaveNet	6549.0	41469.4	20072.0
Benchmark	6626.7	38956.9	19162.7

Tabla 6. Análisis mediana RMSE de todos los modelos según cluster

- Análisis de ranking medio según la totalidad de combinaciones de producto-tienda:

	Cluster 0		Cluster 1		Cluster 2	
	Ranking Medio	# Veces con mejor RMSE	Ranking Medio	# Veces con mejor RMSE	Ranking Medio	# Veces con mejor RMSE
Simple Feed Forward	3.01	42	2.11	82	2.47	65
Temporal Fusion Transformer	2.97	42	2.56	43	2.53	62
DeepAR	3.31	32	3.01	39	2.89	49

WaveNet	2.5	72	3.54	23	3.22	33
Benchmark	3.21	31	3.78	11	3.89	14
Total		219		198		223

Tabla 7. Análisis de ranking medio de todos los modelos según cluster

En conjunto, estos análisis proporcionan una visión integral del desempeño de los modelos, considerando tanto su rendimiento general en términos de error como su consistencia y competitividad al compararlos directamente en las diversas combinaciones de producto y tienda.

4.3. Registro de Tiempo de Entrenamiento

Se midieron los tiempos promedio requeridos para entrenar los modelos de los 10 productos de cada clúster, incluyendo tanto la optimización de hiperparámetros como el entrenamiento final. Las pruebas se realizaron en una computadora MacBook Air (2020) con chip M1, 8 GB de RAM y 8 núcleos de GPU, obteniendo los siguientes resultados:

- Benchmark (Media de ventas): 2 segundos
- Simple FeedForward: 3.764 segundos (~1 hora)
- DeepAR: 59.000 segundos (~16.4 horas)
- Temporal Fusion Transformer (TFT): 51.000 segundos (~14 horas)
- WaveNet: 50.000 segundos (~13.8 horas)

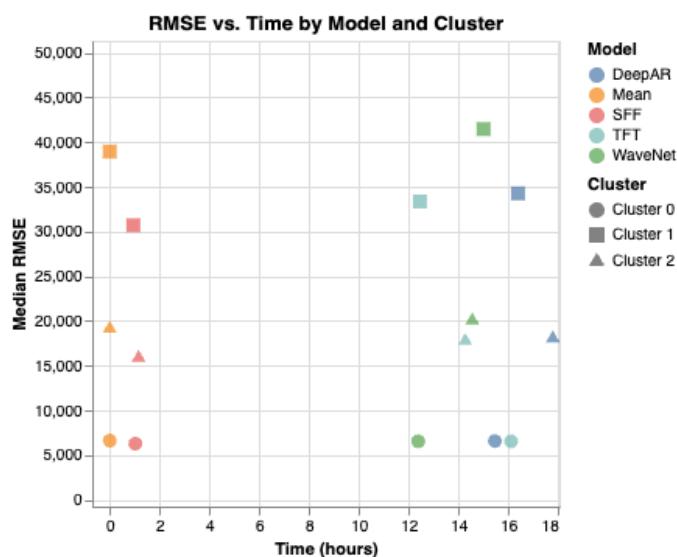


Figura 31. Relación entre RMSE mediano y tiempo de entrenamiento por cluster

Estos valores reflejan el costo computacional de cada enfoque y complementan el análisis de precisión. La configuración del hardware sirve como referencia para reproducibilidad y comparación con otros entornos.

4.4. Análisis de Resultados

Los resultados permiten identificar de forma concluyente al modelo **Simple FeedForward (SFF)** como la mejor alternativa entre todas las evaluadas. Su rendimiento supera sistemáticamente al benchmark en todos los clústeres y, además, logra posicionarse como el modelo con mejor desempeño global cuando se lo compara contra el resto de los modelos.

Comparación contra el benchmark

Al analizar el porcentaje de combinaciones producto-tienda donde SFF logra un menor RMSE que el benchmark, se observa un desempeño superior consistente en todos los clústeres:

- Cluster 0 (productos de baja demanda): SFF supera al benchmark en el 55,7% de los casos, con una mediana de mejora del 6,6% en RMSE.
- Cluster 1 (productos de alta demanda): SFF gana en el 88,9% de los casos, con una mejora mediana del 17,3%, lo que indica una gran capacidad para capturar patrones complejos.
- Cluster 2 (productos de demanda intermedia): obtiene mejores resultados en el 85,2% de los casos, con una mejora mediana del 12,7%.

Estos resultados reflejan una ventaja sólida de SFF respecto al promedio histórico de ventas, especialmente en productos de mayor relevancia comercial (cluster 1).

Comparación entre todos los modelos

Cuando se compara SFF con el resto de los modelos (DeepAR, Temporal Fusion Transformer, WaveNet y benchmark), también se posiciona como el modelo más preciso y consistente en términos globales:

- Cluster 1 (alta demanda): SFF demuestra un desempeño sobresaliente, obteniendo la menor mediana de RMSE (30.709) y siendo el modelo más frecuentemente identificado como el mejor (82 de 198 combinaciones). También alcanza el ranking medio más bajo (2,11), consolidándose como la opción más eficaz para productos críticos con alta demanda.
- Cluster 2 (demanda media): SFF también lidera en este grupo, con la menor mediana de RMSE (15.902) y la mayor cantidad de primeras posiciones (65 de 223 combinaciones). Si bien comparte un desempeño cercano con TFT (ranking medio de 2,47 vs. 2,53), se destaca por su mayor frecuencia de excelencia, lo que refuerza su solidez en escenarios de demanda intermedia.
- Cluster 0 (baja demanda y alto ruido): En este grupo, la competencia es más ajustada. WaveNet obtiene más primeras posiciones (72 vs. 42 de SFF) y un mejor ranking medio (2,5 vs. 3,01). Sin embargo, SFF consigue la mejor mediana de RMSE (6.287), lo que sugiere una mayor estabilidad frente a valores atípicos. Su rendimiento sigue siendo competitivo, mostrando una buena adaptabilidad incluso en contextos más desafiantes.

Evaluación global y ranking final

Integrando los resultados por clúster, puede establecerse un ranking general de modelos basado en precisión, estabilidad y número de victorias:

1. Simple FeedForward (SFF): Mejor desempeño general. Líder en precisión (mediana de RMSE), frecuencia de victorias y ranking promedio. Además, presenta una eficiencia computacional significativamente superior, con tiempos de entrenamiento mucho más bajos que modelos como TFT o WaveNet, lo que lo convierte en una opción altamente viable para entornos productivos donde los recursos de cómputo son un factor clave.
2. Temporal Fusion Transformer (TFT): Segunda mejor alternativa. Compite muy de cerca con SFF en cluster 2 y se mantiene fuerte en cluster 1, aunque su alto costo computacional limita su aplicabilidad en escenarios con restricciones de recursos.
3. WaveNet: Buen rendimiento en cluster 0, pero desempeño más irregular en el resto. Su ventaja en ciertos contextos específicos puede justificar su uso como complemento.
4. DeepAR: Resultados intermedios. Mejora frente al benchmark, especialmente en cluster 2, pero no logra destacar frente a los demás modelos en rankings globales ni en métricas agregadas.
5. Benchmark (promedio histórico): Punto de comparación base. Su simplicidad y bajo costo computacional son sus principales ventajas, pero queda sistemáticamente superado por todos los modelos basados en aprendizaje automático.

4.5 Comparación Mejor Modelo vs XGBoost

Una vez identificado el modelo de mejor desempeño —el modelo Simple FeedForward (SFF)— se procedió a compararlo con XGBoost siguiendo la misma metodología de entrenamiento.

Para cada clúster, se evaluó qué modelo obtuvo el menor RMSE en todas las combinaciones producto - tienda.

Mejor RMSE según combinaciones producto - tienda		
	Simple FeedForward	XBGooost
Cluster 0	133 - (60.7%)	86 - (39.3%)
Cluster 1	148 - (74.7%)	50 - (25.3%)
Cluster 2	131 - (58.7%)	92 - (41.3%)

Tabla 8. Comparación del RMSE entre Simple FeedForward y XGBoost por cluster

Complementariamente, se calculó la mediana del RMSE para ambos modelos en cada clúster. Los resultados se resumen en la siguiente tabla:

	Cluster 0 Mediana MRSE	Cluster 1 Mediana MRSE	Cluster 2 Mediana MRSE
SimpleFeed Forward	6287.0	30709.5	15902.0
XGBoost	6921.8	40400.3	18007.9

Tabla 9. Comparación de la mediana del RMSE entre SFF y XGBoost por cluster

Los valores presentados evidenciaron que el modelo SFF no solo logró más “victorias” en la comparación directa, sino que también mantuvo un error mediano sistemáticamente inferior al de XGBoost en los tres clústeres. Esta superioridad fue especialmente pronunciada en el Clúster 1, donde la diferencia en la cantidad de combinaciones ganadas por SFF frente a XGBoost fue la más amplia entre los tres clústeres evaluados.

En conclusión, el modelo de red neuronal Simple FeedForward demostró un rendimiento notablemente más preciso que XGBoost, consolidándose como una alternativa más efectiva para tareas de predicción de demanda a nivel granular. Este resultado respalda su selección como modelo final a implementar en el sistema de Smart Replenishment.

4.6 Análisis de Hiperparámetros del Modelo Simple FeedForward

A continuación, se presenta un análisis de los hiperparámetros óptimos de arquitectura y contexto utilizados en los modelos Simple FeedForward en los diferentes clusters analizados.

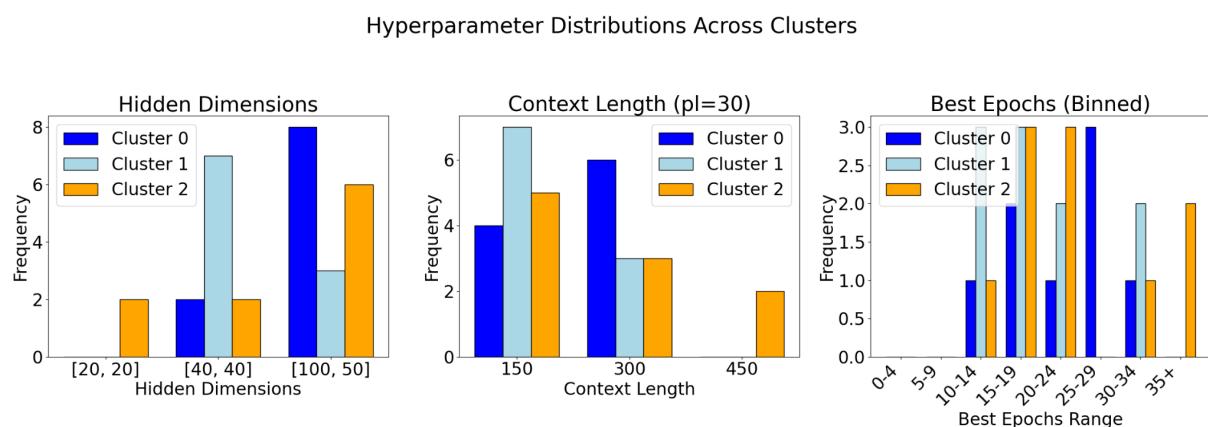


Figura 32. Hiperparámetros utilizados por Simple FeedForward

El análisis de los hiperparámetros según cluster revela tendencias interesantes en la configuración del modelo y el proceso de entrenamiento:

- **Bajo Volumen (Cluster 0):** Muestra modelos más complejos (dimensiones ocultas grandes) y variedad en la cantidad de datos históricos usados para predecir (longitud del contexto variable). El número de veces que se entrena el modelo (épocas óptimas) no tiene un patrón claro.
- **Alto Volumen (Cluster 1):** Muestra modelos más simples (dimensiones ocultas pequeñas), usa menos datos históricos recientes para predecir (contexto corto) y necesita menos entrenamiento (épocas óptimas bajas).
- **Volumen Medio (Cluster 2):** Muestra una mezcla de complejidad de modelo y cantidad de datos históricos usados. El entrenamiento requerido se sitúa en un punto intermedio.

En los datos analizados, el volumen de ventas influye en la configuración ideal del modelo de forecasting. Se observa la tendencia de que a mayor volumen, los modelos óptimos tienden a ser más simples y requieren menos entrenamiento. Por el contrario, a menor volumen, se favorecen modelos más complejos y existe una mayor variabilidad en la cantidad de datos históricos utilizados para realizar las predicciones.

5. Conclusiones

Este trabajo presentó una solución eficaz y escalable para el pronóstico de demanda en contextos de alta granularidad dentro de la industria del retail. A través del diseño de una arquitectura basada en modelos de deep learning entrenados a nivel de producto, fue posible capturar patrones generales de comportamiento considerando simultáneamente las series temporales de todas las tiendas asociadas a cada artículo. Esta estrategia no solo incrementó de forma significativa la precisión de las predicciones, sino que también permitió una reducción sustancial en los tiempos de entrenamiento, evitando la necesidad de desarrollar un modelo específico para cada combinación producto-tienda.

Entre los modelos evaluados, el enfoque basado en una red neuronal Simple FeedForward (SFF) se consolidó como la alternativa más robusta. Este modelo superó sistemáticamente al benchmark tradicional y a XGBoost —uno de los algoritmos más utilizados en la industria— en todos los clusters analizados, mostrando un rendimiento especialmente destacado en productos de alta demanda. Su bajo costo computacional y su alta capacidad de generalización lo convierten en una opción especialmente atractiva para entornos productivos donde la eficiencia y la escalabilidad son factores determinantes.

En este contexto, se proponen dos recomendaciones concretas:

1. **Incorporación del modelo Simple FeedForward al sistema Smart Replenishment:** dada su solidez empírica, se sugiere integrar este modelo como componente central del sistema de predicción de demanda. Su desempeño consistente en términos de precisión, velocidad de entrenamiento y estabilidad lo convierte en una herramienta confiable para generar pronósticos masivos. Su adopción permitiría anticipar la demanda con mayor exactitud, mejorando la planificación del inventario, reduciendo quiebres de stock y excesos de mercadería, y optimizando así los resultados operativos y comerciales del negocio.
2. **Adopción de una estrategia de modelado a nivel de producto para predicción a gran escala:** como segunda contribución clave de este trabajo, se recomienda formalmente el uso de una arquitectura de predicción a nivel de producto como metodología ideal para abordar desafíos de forecasting masivo en retail. Esta estrategia demostró ser no solo más eficiente desde el punto de vista computacional, sino también más precisa y sostenible a medida que el sistema crece en complejidad. Representa un enfoque replicable, adaptable y alineado con las necesidades reales de entornos productivos que requieren soluciones de pronóstico dinámicas, precisas y escalables.

En conjunto, estos resultados evidencian el valor estratégico de la solución propuesta y refuerzan su potencial de impacto en aplicaciones reales, contribuyendo significativamente a la toma de decisiones basadas en datos en el ámbito del retail.

6. Futuras Direcciones

A partir de los resultados obtenidos, se identifican varias líneas de investigación con potencial para fortalecer y expandir la solución propuesta. Si bien este trabajo se enfocó principalmente en el desarrollo y evaluación de arquitecturas de modelado, existen oportunidades relevantes en áreas complementarias que podrían llevar el sistema a un nuevo nivel de desempeño y adaptabilidad:

1. **Incorporación de variables exógenas:** Dado que el objetivo principal de esta tesis fue explorar estrategias de modelado escalables, el diseño de atributos se mantuvo deliberadamente acotado. Sin embargo, enriquecer las entradas del modelo con variables externas podría aportar contexto adicional y mejorar la precisión de las predicciones. Factores globales como feriados, promociones o estacionalidades, así como características específicas de cada tienda (ubicación, tamaño, tipo de clientela, etc.), pueden ayudar a capturar influencias que no están presentes en la serie histórica.
2. **Optimización de arquitecturas complejas y régimen de entrenamiento:** Modelos más sofisticados como DeepAR, WaveNet o Temporal Fusion Transformer mostraron un rendimiento competitivo en ciertos escenarios. Explorar versiones más profundas o ajustadas de estas arquitecturas, incorporando mecanismos como atención temporal o capas adicionales, podría permitir una mejor captura de dependencias complejas. A su vez, el régimen de entrenamiento fue restringido por limitaciones de cómputo; contar con mayores recursos permitiría aumentar el número de épocas y realizar búsquedas de hiperparámetros más exhaustivas, lo cual podría mejorar la capacidad de generalización de los modelos sin incurrir en sobreajuste.
3. **Agrupamiento de tiendas para escalar el enfoque:** En escenarios donde un mismo producto comienza a comercializarse en una cantidad muy grande de tiendas, el enfoque actual podría verse limitado por la complejidad del modelo resultante. Como solución, se propone aplicar técnicas de aprendizaje no supervisado —por ejemplo, clustering con K-Means— para agrupar las tiendas en subconjuntos homogéneos según sus patrones de demanda específicos para cada producto. Esto permitiría mantener el enfoque de modelado a nivel de producto, pero dividiendo el entrenamiento en múltiples modelos por grupo de tiendas, lo que conservaría la escalabilidad del sistema sin perder precisión ni capacidad de adaptación.

Estas líneas de trabajo representan oportunidades claras para evolucionar hacia soluciones más robustas, precisas y alineadas con las demandas dinámicas del entorno retail, manteniendo el foco en la eficiencia, la escalabilidad y la aplicabilidad en contextos reales de negocio.

Referencias

Alexandrov A, Benidis K, Bohlke-Schneider M, Flunkert V, Gasthaus J, Januschowski T, Smola AJ (2019) GluonTS: Probabilistic Time Series Models in Python. arXiv. <https://arxiv.org/abs/1906.05264>

Bergstra J, Bengio Y (2012) Random Search for Hyper-Parameter Optimization. Journal of Machine Learning Research. <http://www.jmlr.org/papers/v13/bergstra12a.html>

Bergstra J, Bardenet R, Bengio Y, Kégl B (2011) Algorithms for Hyper-Parameter Optimization. Advances in Neural Information Processing Systems. <https://papers.nips.cc/paper/2011/hash/86e8f7ab32cf12577bc2619bc635690-Abstract.html>

Borovykh A, Bohte S, Oosterlee CW (2017) Conditional Time Series Forecasting with Convolutional Neural Networks. arXiv. <https://arxiv.org/abs/1703.04691>

Chen T, Guestrin C (2016) XGBoost: A Scalable Tree Boosting System. arXiv. <https://arxiv.org/abs/1603.02754>

GluonTS (s f) SimpleFeedForwardEstimator.
https://ts.gluon.ai/stable/api/gluonts/gluonts.torch.model.simple_feedforward.estimator.html

GluonTS (s f) Temporal Fusion Transformer Estimator.
<https://ts.gluon.ai/dev/api/gluonts/gluonts.torch.model.tft.estimator.html>

GluonTS (s f) DeepAREstimator.
<https://ts.gluon.ai/dev/api/gluonts/gluonts.torch.model.deepar.estimator.html>

GluonTS (s f) WaveNetEstimator.
<https://ts.gluon.ai/dev/api/gluonts/gluonts.torch.model.wavenet.estimator.html>

Hyndman RJ, Athanasopoulos G (2021) Forecasting: Principles and Practice (3rd ed). OTexts. <https://otexts.com/fpp3/>

Lim B, Arik SO, Loeff N, Pfister T (2021) Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting. arXiv. <https://arxiv.org/abs/1912.09363>

Qin Y, Song D, Chen H, Cheng W, Jiang G, Cottrell G (2017) A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction. arXiv. <https://arxiv.org/abs/1704.02971>

Salinas D, Flunkert V, Gasthaus J, Januschowski T (2020) DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks. arXiv. <https://arxiv.org/abs/1704.04110>

Van den Oord A, Dieleman S, Zen H, Simonyan K, Vinyals O, Graves A, Kalchbrenner N, Senior A, Kavukcuoglu K (2016) WaveNet: A Generative Model for Raw Audio. arXiv.

<https://arxiv.org/abs/1609.03499>

Watanabe S (2023) Tree-Structured Parzen Estimator: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance. arXiv.

<https://arxiv.org/abs/2304.11127>

Wang J, Li Y, Luo H (2023) Variational Mode Decomposition Enhanced Attention-Based Forecasting for Multivariate Time Series. MDPI Mathematics.

<https://www.mdpi.com/2227-7390/11/1/13>

Zhang J, Wang P, Li X, Wang J (2022) A Hybrid Ensemble Forecasting Framework Based on Statistical, Machine Learning and Deep Learning Models for Sales Demand Forecasting. Procedia Computer Science. <https://doi.org/10.1016/j.procs.2022.01.173>