Pronóstico de Demanda de Huevos para empresa avícola mediante Machine Learning

Maestría en Inteligencia Artificial y Ciencia de Datos
Asignatura: Aprendizaje Automático
Profesor: Francisco Jose Mercado Rivera, PhD
de junio de 2025

Entrega final de proyecto

Juan David Díaz Calero Cód. 22502473 juan.diaz_c@uao.edu.co
Carlos Andres Becerra Cód. 22500215 carlos.becerra@uao.edu.co
Luis Carlos Correa Zuluaga Cód. 22501541 luis_carlos.correa@uao.edu.co

Resumen

En este proyecto, sugerimos implementamos tres modelos de machine learning SARIMA, Prophet (Facebook), Random Forest y XGBoost en stacking, para predecir las ventas de huevos de una compañia avicola. Se entrenaron los modelos con un dataset al cual se le hizo el EDA que fue limpiado previamente.

Index Terms

Machine Learning, Pronóstico de Demanda, Huevos, Productos Perecederos, Series Temporales, SARIMA, Prophet (Facebook), Random Forest, XGBoost, Ridge Regression, Stacking, Optimización Logística, Reducción de Desperdicios, Análisis de Datos, Ingeniería de Características, ETL (Extracción, Transformación y Carga de Datos), MAPE (Mean Absolute Percentage Error), MAE (Mean Absolute Error), RMSE (Root Mean Squared Error), Accuracy, Precision, Recall, F1-Score, Empresa Avícola, Colombia, Valle del Cauca, Pronóstico de ventas

I. Introducción: Descripción del fenómeno / proceso modelado y el problema a abordar

El fenómeno o proceso modelado en este proyecto es el pronóstico de la demanda de huevos. Este proceso se enfoca en la actividad de una empresa avícola regional ubicada en el Valle del Cauca, Colombia, que se dedica a la producción y comercialización de huevos frescos y tiene más de 50 años de trayectoria. La empresa distribuye sus productos a través de canales de venta como Autoservicios, TAT (Tienda a Tienda), Horeca (Hoteles, Restaurantes, Cafeterías) y mayoristas en todo el territorio colombiano. La literatura subraya que, para bienes perecederos como los huevos, la precisión en el pronóstico es de ïmportancia crítica", ya que los errores pueden conducir directamente a pérdidas económicas y afectar la satisfacción del cliente. La demanda de huevos, en particular, presenta desafíos complejos debido a su vida útil limitada, fluctuaciones estacionales pronunciadas, sensibilidad a eventos externos y promociones, y variabilidad en la calidad del producto. Además, el consumo per cápita de huevo en Colombia ha ido en aumento, posicionando el producto como uno de los más importantes en la canasta familiar y llevando a más avicultores a incrementar su producción, lo que puede afectar el precio (oferta y demanda).

El problema central que aborda este proyecto radica en que la empresa avícola, a pesar de su larga trayectoria, utiliza métodos de pronóstico tradicionales, como promedios históricos y estimaciones manuales. Estos métodos resultan en una desviación promedio significativa del 20-25 % respecto a la demanda real.

Las principales consecuencias de esta imprecisión en el pronóstico son:

- Subproducción y pérdida de ventas: En picos de demanda no anticipados, la empresa calcula una pérdida de ventas del 5-7 % y experimenta dificultades para cumplir con los pedidos de clientes clave.
- Ineficiencias operativas y costos elevados: La imprecisión lleva a la empresa a incurrir en costos elevados de transporte urgente para cubrir los déficits de producto y resulta en una asignación subóptima de recursos tanto en las granjas como en los centros de distribución.

El objetivo fundamental del proyecto es desarrollar y evaluar un modelo de Machine Learning que permita mejorar la precisión de las predicciones de demanda a un error inferior al 10 % (MAPE). Con esto, la empresa espera lograr una reducción de desperdicios en al menos un 20 %, mejorar la eficiencia logística y optimizar los costos de inventario y distribución, fortaleciendo así su posición competitiva en el mercado.

II. Solución propuesta

La solución propuesta abarca un proceso de Extracción, Transformación y Carga (ETL) de datos, que incluye la unificación de formatos, detección y corrección de valores atípicos o faltantes, y la generación de variables de calendario y retraso (lags). Se explorarán y entrenarán diversos modelos de Machine Learning, incluyendo SARIMA, Prophet (de Facebook), Random Forest, XGBoost, y la combinación de modelos a través de Stacking utilizando Ridge Regression o Linear Regression como metamodelo.

La evaluación de los modelos se realizará utilizando métricas cuantitativas como MAPE (Mean Absolute Percentage Error), MAE (Mean Absolute Error), y RMSE (Root Mean Squared Error), junto con un análisis cualitativo visual de las predicciones. Los resultados esperados incluyen una reducción de desperdicios en al menos un 20 %, mejora de la eficiencia logística, optimización de costos de inventario y distribución, y el fortalecimiento de la posición competitiva de la empresa.

III. Proceso de obtención y generación del dataset

El proceso de obtención y generación del dataset para este proyecto se realizó a través de varias fases clave, que incluyó la comprensión de las fuentes de datos, la extracción, la transformación (ETL) y la validación de los datos:

- Fuente de Datos:
  - El historial de ventas abarca dos años (2022-2023) para el objetivo principal de pronosticar la demanda de huevos. Sin embargo, otras secciones mencionan el periodo de Enero 2023 a Diciembre 2024 (24 meses) para el historial de ventas.
  - Los datos se extraen de SAP R/3 y planillas Excel complementarias.
  - En SAP R/3, se identificaron tablas clave como:
    - KNA1: Datos generales del maestro de clientes (nombre, NIT o cédula, dirección, teléfono).
    - KNVV: Información específica de ventas del cliente (tipo de cliente, canal de distribución, zona de ventas, vendedor, condiciones de pago).
    - MAKT: Descripciones de los productos.
    - VBRK: Datos de cabecera de las facturas (fechas, cliente, destinatario, canal de distribución, totales).
    - VBRP: Datos de posición de las facturas (cantidad, producto, descuentos, precio).
  - Las variables clave obtenidas incluyen la fecha de venta, cantidad vendida diaria, tipo de huevo, precio promedio, e indicadores de festivos y quincenas.
  - La muestra de datos muestra_de_dataset.txt incluye campos como factura, cantidad_neta, fecha_de_factura, nomb_material (nombre del material como "HUEVO ORO ROJO B SUELTO.º "HUEVOS ORO PLUS AAA X 30"), TIPO (como "SUELTO.º .ESTUCHERIA"), y CIUDAD (como "MEDELLIN").
- Extracción de Datos:
  - Para la extracción directa desde SAP, la empresa utiliza la herramienta SAP Crystal Reports, que permite consultar múltiples tablas de SAP simultáneamente y crear modelos relacionales entre ellas.
  - La información generada por Crystal Reports puede descargarse en formatos como XLS, XLSX o CSV.
  - La extracción de datos mediante Crystal Reports se planifica como una carga incremental diaria para asegurar la actualización continua sin duplicación.
  - Los archivos Excel de la fuerza de ventas se consolidan para permitir la comparación con las ventas reales mensuales.
- Transformación de Datos (ETL):
  - La transformación se realiza en dos etapas: primero en SAP Crystal Reports para organizar y relacionar las tablas extraídas, y luego en Python para limpieza, estandarización y consolidación.
  - Los procesos en Python incluyen:
    - Eliminación de valores nulos, registros duplicados y datos inconsistentes. Por ejemplo, se convierte cantidad_neta a numérico y se eliminan filas con valores nulos en esta columna. También se descartan valores negativos de cantidad_neta si no corresponden a devoluciones.
    - Estandarización de formatos de fecha y texto, asegurando consistencia en nombres de columnas, formatos de fechas, montos y códigos de identificación.
    - Agregación de ventas diarias a semanales. El código muestra cómo las ventas diarias se agrupan por cliente y ciudad para sumar la cantidad_neta.
    - Generación de variables de calendario (mes, año, festivos) y variables de retraso (lags) y ventanas móviles.
    - Transformación de datos mediante agregación a niveles mensuales por cliente y por tipo de empaque.
    - Análisis de estacionariedad con pruebas como Dickey-Fuller Aumentada.

- Automatización de los scripts de transformación para asegurar eficiencia y replicabilidad en cada actualización.
- Protección de información confidencial mediante el reemplazo de nombres de clientes, zonas de ventas y canales de distribución por identificadores genéricos (Cliente 1, Cliente 2, etc.).

- Validación de Datos:
  - Se realizan validaciones para asegurar la confiabilidad de los datos antes de integrarlos en Power BI. Esto incluye:
    - Revisión de consistencia comparando los datos transformados con los reportes de ventas mensuales (ventas totales, cantidad de unidades vendidas, cantidad de clientes facturados).
    - Verificación de claves únicas para identificadores de clientes, productos y vendedores.
    - Detección de valores inusuales (ej. precios fuera de rango, nombres de productos inconsistentes) utilizando Python.
    - Revisión de la correcta anonimización de datos confidenciales (clientes, zonas de ventas, canales de distribución).
    - Automatización de los controles de validación mediante scripts en Python.
    - Generación de un reporte automático de anomalías detectadas para su corrección.

  finalmente se tienen dos archivos principales: 2022_limpio.csv y 2023_limpio.csv. Estos archivos contienen el dataset completo y listo para el entrenamiento de los modelos, que suma un total de 1,295,578 registros.

IV. Descripción del problema de aprendizaje automáticos

El problema de aprendizaje automático abordado en este proyecto es un problema de Regresión de Series Temporales.

V. ¿Qué tipo de problema es?

- Es un problema de regresión porque el objetivo es predecir una variable continua y numérica: la cantidad diaria y/o semanal de huevos que se demandará. No se busca una clasificación (categoría) sino un valor específico.
- Más específicamente, es un problema de pronóstico de series temporales. Esto se debe a que la predicción de la demanda se basa en datos históricos ordenados cronológicamente. La demanda de huevos es un fenómeno que exhibe patrones de estacionalidad (fluctuaciones pronunciadas), tendencia, y autocorrelación, características intrínsecas de las series de tiempo. Los huevos son un producto perecedero, lo que hace que la precisión del pronóstico sea de ïmportancia críticaz un "desafío complejo"debido a su vida útil limitada y la volatilidad de la demanda.

VI. ¿Cuál es la tarea de aprendizaje que se debe realizar?

La tarea de aprendizaje es desarrollar y evaluar un modelo de Machine Learning para pronosticar la demanda de huevos. Esto implica:

- Aprender patrones y relaciones: Los modelos deben aprender de un dataset histórico de ventas de dos años (2022-2023 o Enero 2023 - Diciembre 2024), que incluye variables clave como la fecha de venta, cantidad vendida diaria, tipo de huevo, precio promedio, e indicadores de festivos y quincenas. El aprendizaje automático permitirá a los modelos capturar relaciones no lineales y dependencias complejas en los datos, algo que los métodos tradicionales de la avícola (promedios históricos y estimaciones manuales) no logran con precisión.
- Mejorar la precisión del pronóstico: El objetivo principal es reducir la desviación promedio actual del 20-25 % a un error inferior al 10 % (MAPE). Para ello, se entrenarán y evaluarán diversos modelos de Machine Learning, incluyendo SARIMA, Prophet (de Facebook), Random Forest, XGBoost, y Stacking (combinación de modelos, utilizando Ridge Regression o Linear Regression como metamodelo). La literatura también sugiere modelos como LSTM y SVR para productos perecederos.
- Optimizar las operaciones empresariales: Al mejorar la precisión del pronóstico, la tarea de aprendizaje busca directamente mitigar problemas como la subproducción (pérdida de ventas del 5-7 %), los costos elevados en transporte urgente, y la asignación subóptima de recursos. Los resultados esperados incluyen una reducción de desperdicios en al menos un 20 %, mejora de la eficiencia logística y optimización de costos de inventario y distribución.

VII. Preprocesamiento de datos

VII-A. Limpieza de Datos

Esta fase implica la eliminación de valores nulos, registros duplicados y datos inconsistentes. Específicamente, se convierte la columna cantidad_neta a un formato numérico y se eliminan las filas con valores nulos en esta columna. También se descartan los valores negativos de cantidad_neta si no corresponden a devoluciones. Además, se lleva a cabo la detección y corrección de valores atípicos con imputación.

VII-B. Transformación y Agregación de Datos

- Estandarización de formatos: Se unifican las estructuras de datos, asegurando consistencia en nombres de columnas, formatos de fechas, montos y códigos de identificación.
- Anonimización: Para proteger la información confidencial, se reemplazan los nombres de clientes, zonas de ventas y canales de distribución con identificadores genéricos.

VII-C. Ingeniería de Características

Se crean nuevas variables a partir de los datos existentes para mejorar el rendimiento del modelo Arboles binarios-XGBoost apilados. Esto incluye:

- Generación de variables de calendario (mes, año, festivos).
- Creación de variables de retraso (lags) y ventanas móviles.

VII-D. Escalado de Datos

Los datos son escalados utilizando StandardScaler() antes de entrenar el modelo de Arboles binarios-XGBoost apilados.

VIII. Entrenamiento y evaluación de tres modelos

VIII-A. Modelo SARIMA

El modelo SARIMA (Seasonal AutoRegressive Integrated Moving Average) es un modelo estadístico clásico ampliamente utilizado para el pronóstico de series temporales. Es particularmente efectivo para capturar autocorrelación y estacionalidad en los datos, lo cual es relevante para series temporales agrícolas.

VIII-A1. Carga y Concatenación de Datos:

- Se cargaron los datos históricos de ventas de los años 2022 y 2023 desde archivos CSV separados (2022_limpio.csv y 2023_limpio.csv).
- Estos DataFrames se concatenaron para formar un único conjunto de datos (df).

VIII-A2. Limpieza de Datos:

- Se aseguró que la columna cantidad_neta fuera numérica, convirtiendo los valores a numéricos y reemplazando los errores con NaN (valores nulos).
- Se eliminaron las filas que contenían valores nulos en la columna cantidad_neta.

VIII-A3. Agrupación de Ventas Diarias por Cliente y Ciudad:

- Las ventas diarias se agruparon por fecha_de_factura, solicitante (cliente) y CIUDAD.
- Se sumó la cantidad_neta para cada una de estas agrupaciones para obtener las ventas diarias consolidadas.
- Las columnas resultantes se renombraron a fecha, solicitante, CIUDAD, y cantidad_neta.

VIII-A4. Filtrado de Clientes y Ciudades con Suficientes Datos:

- Se identificaron las combinaciones únicas de solicitante y CIUDAD.
- Solo se seleccionaron aquellas combinaciones que tuvieran al menos 180 días de datos (observaciones) para asegurar una cantidad suficiente de información para el entrenamiento y la prueba del modelo.

VIII-A5. Definición de la Fecha de Corte:

- Se estableció una fecha de corte (2023-07-01) para dividir los datos en conjuntos de entrenamiento y prueba.

VIII-A6. Bucle de Evaluación por Cliente y Ciudad:

- Se creó un bucle para iterar a través de cada combinación filtrada de cliente y ciudad.
- Para cada combinación, se filtró el subconjunto de datos relevante y se ordenó por fecha.
- El índice del DataFrame se estableció como la columna fecha y se rellenaron los días faltantes con 0 para asegurar una frecuencia diaria continua.

VIII-A7. División en Conjuntos de Entrenamiento y Prueba:

- El conjunto de entrenamiento (train) incluyó los datos anteriores a la fecha_corte.
- El conjunto de prueba (test) incluyó los datos a partir de la fecha_corte.
- Se agregó una condición para saltar el entrenamiento si no había suficientes datos en los conjuntos de entrenamiento (menos de 180 días) o prueba (menos de 30 días).

VIII-A8. Entrenamiento del Modelo SARIMA:

- Dentro del bucle, para cada subconjunto de datos, se entrenó un modelo SARIMAX de statsmodels.tsa.statespace.sarimax.
- Los parámetros específicos utilizados para SARIMA fueron: order=(1, 1, 1) y seasonal_order=(1, 1, 1, 7). El componente estacional (1,1,1,7) indica una estacionalidad semanal (período de 7 días) el mas bajo posible para el modelo SARIMA de la libreria *statsmodels*.
- El modelo se ajustó con model.fit(disp=False).

VIII-A9. Generación de Pronósticos:

- Una vez entrenado el modelo, se generaron las predicciones (pred) para el período de tiempo del conjunto de prueba, desde la primera hasta la última fecha del test.

VIII-A10. Evaluación del Modelo:

- Se calcularon las métricas de evaluación clave:
  - MAE (Mean Absolute Error): Para medir el error absoluto promedio entre las predicciones y los valores reales.
  - Promedio de Ventas Reales: El promedio de las ventas en el conjunto de prueba.
  - MAE Relativo: MAE dividido por el promedio de ventas reales.
  - Accuracy, Precision, Recall, y F1-score: Calculadas como 1 - (MAE / Promedio_real) y sus derivadas, donde un valor más cercano a 1 indica mejor desempeño.
  - MAPE (Mean Absolute Percentage Error): Con el objetivo de ser inferior al 10 %. Para su cálculo, se filtraron los valores reales que no fueran cero para evitar divisiones por cero.
- Los resultados de cada combinación (cliente, ciudad y sus métricas) se almacenaron en una lista llamada resultados.

VIII-A11. Visualización de Resultados:

- Se generaron gráficos de series temporales que comparan las ventas reales con las predicciones, así como gráficos de dispersión (scatter plots) de las predicciones frente a las ventas reales.
- Estas visualizaciones se guardaron en una carpeta plots.

VIII-A12. Guardado de Resultados:

- Finalmente, los resultados de la evaluación de todos los clientes y ciudades se consolidaron en un DataFrame y se guardaron en un archivo Excel llamado resultados_modelo_sarima.xlsx.

VIII-B. Modelo Prophet

El modelo Prophet es una librería de código abierto desarrollada por Facebook, diseñada específicamente para el pronóstico de series temporales, particularmente aquellas que exhiben múltiples estacionalidades y la influencia de festivos. Es reconocido por su robustez ante datos faltantes y valores atípicos (outliers). Esta característica lo hace especialmente adecuado para el pronóstico de demanda en negocios, como la industria avícola, donde la volatilidad de la demanda, la estacionalidad (diaria, semanal, etc.) y los eventos externos pueden influir en las ventas.

VIII-B1. Carga y Concatenación de Datos:

- Se inició cargando los datos históricos de ventas de los años 2022 y 2023 desde archivos CSV (2022_limpio.csv y 2023_limpio.csv).
- Estos DataFrames se concatenaron para formar un único conjunto de datos (df), asegurándose de que la columna fecha_de_factura fuera parseada como fecha y que el formato dayfirst=True se aplicara.

VIII-B2. Preparación y Limpieza de Datos:

- Se aseguró que la columna cantidad_neta fuera numérica y se agruparon las ventas diarias por fecha_de_factura (renombrada a ds para Prophet), solicitante (cliente) y CIUDAD. Se sumó la cantidad_neta (renombrada a y para Prophet) para obtener las ventas diarias consolidadas.
- Se identificaron y filtraron las combinaciones únicas de solicitante y CIUDAD que tuvieran al menos 180 días de datos para asegurar suficiente información para el modelado.

VIII-B3. Definición de la Fecha de Corte:

- Se estableció una fecha de corte (2023-07-01) para dividir los datos en conjuntos de entrenamiento y prueba. El conjunto de entrenamiento abarcó 18 meses, y el de prueba 6 meses.

VIII-B4. Bucle de Evaluación por Cliente y Ciudad:

- Se implementó un bucle para iterar a través de cada combinación filtrada de cliente y ciudad.
- Dentro del bucle, se filtró el subconjunto de datos relevante para cada combinación y se ordenó por fecha.
- Se verificó que hubiera suficiente información para la prueba; si el conjunto de entrenamiento tenía menos de 100 días o el de prueba menos de 30 días, se saltaba esa combinación.

VIII-B5. Entrenamiento del Modelo Prophet:

- Para cada subconjunto de datos (entrenamiento), se inicializó y entrenó un modelo Prophet. Se configuró daily_seasonality=True para capturar patrones diarios de estacionalidad.
- El modelo se ajustó a los datos de entrenamiento (train) utilizando modelo.fit(train).

VIII-B6. Generación de Pronósticos:

- Una vez entrenado, se utilizó modelo.make_future_dataframe para crear un DataFrame con las fechas futuras correspondientes al período del conjunto de prueba, con una frecuencia diaria (freq='D').
- Las predicciones (yhat) se generaron utilizando modelo.predict(future).

VIII-B7. Evaluación del Modelo:

- Las predicciones se compararon con los valores reales del conjunto de prueba (test).
- Se calcularon métricas clave de evaluación:
  - MAE (Mean Absolute Error): Para medir el error absoluto promedio.
  - Promedio de Ventas Reales: El promedio de las ventas en el conjunto de prueba.
  - MAE Relativo: MAE dividido por el promedio de ventas reales.
  - Accuracy, Precision, Recall y F1-score: Calculadas como 1 - (MAE / Promedio_real) y sus derivadas, buscando valores más cercanos a 1 para un mejor desempeño.
  - MAPE (Mean Absolute Percentage Error): Se calculó específicamente filtrando los valores reales que no fueran cero para evitar divisiones por cero. El objetivo del proyecto era reducir el MAPE a menos del 10 %.
- Los resultados de cada combinación (cliente, ciudad y sus métricas) se almacenaron en una lista llamada resultados.

VIII-B8. Visualización de Resultados:

- Se generaron gráficos de series temporales comparando las ventas reales con las predicciones, y gráficos de dispersión (scatter plots) de las predicciones frente a las ventas reales.
- Estas visualizaciones se guardaron en una carpeta plots para un análisis visual detallado del rendimiento del modelo por cada cliente y ciudad.

VIII-C. Modelos Random Forest y XGBoost apilados

Los modelos Random Forest y XGBoost son algoritmos de aprendizaje automático ampliamente utilizados, especialmente efectivos para el pronóstico de series temporales y para capturar relaciones complejas en los datos.

VIII-C1. ¿Qué son los modelos Random Forest y XGBoost?:

- Random Forest (Bosque Aleatorio): Es un algoritmo de aprendizaje supervisado que construye múltiples árboles de decisión durante el entrenamiento y genera la salida que es el modo de las clases (para clasificación) o la predicción media (para regresión) de los árboles individuales. En el contexto del pronóstico de demanda, ha demostrado ser efectivo para capturar relaciones no lineales y manejar diversas características. También es útil para la selección de variables al evaluar la importancia de las características.
- XGBoost (Extreme Gradient Boosting): Es una implementación optimizada y distribuida de algoritmos de gradient boosting. Es reconocido por su precisión y velocidad, siendo uno de los algoritmos más precisos para datos complejos. Al igual que Random Forest, es efectivo para capturar relaciones no lineales y manejar diversas características. Se ha demostrado que XGBoost supera a Random Forest en la predicción de la tasa de producción agricola, destacando su capacidad para capturar patrones sutiles en datos no lineales. Además, en comparaciones con modelos de Deep Learning como LSTM para pronosticar alimentos perecederos, XGBoost a veces ha demostrado superar a LSTM en RMSE a nivel de tienda.

VIII-C2. ¿Por qué se pueden ensamblar (apilar) como se hizo?: Los modelos se pueden ensamblar o .ªpilar"(stacking) porque combinando las predicciones de varios modelos base (como Random Forest y XGBoost) a través de un "meta-modelo", se busca aprovechar las fortalezas de cada algoritmo y mitigar sus debilidades, lo que puede llevar a una mejora significativa en la precisión general del pronóstico.

Un enfoque de combinación óptima de modelos (ensemble) ha demostrado mejorar la exactitud en el pronóstico de ventas hasta en un 12 % en comparación con el mejor modelo individual. Al combinar modelos como Random Forest y XGBoost, que son buenos para relaciones no lineales, el modelo apilado puede generar predicciones más robustas y precisas.

VIII-C3. Carga y Concatenación de Datos:

- Se cargaron los datos históricos de ventas de los años 2022 y 2023 desde archivos CSV (2022_limpio.csv y 2023_limpio.csv). El dataset completo tiene 1,295,578 registros.
- Se concatenaron estos datos en un único DataFrame.

VIII-C4. Limpieza Básica de Datos:

- La columna cantidad_neta se convirtió a tipo numérico, manejando posibles errores.
- Se eliminaron las filas que contenían valores nulos (NaN) en cantidad_neta.
- Se descartaron valores negativos en cantidad_neta si no correspondían a devoluciones.

VIII-C5. Agrupación de Ventas Diarias por Cliente y Ciudad:

- Las ventas se agruparon por fecha_de_factura, solicitante (cliente) y CIUDAD, sumando la cantidad_neta para obtener las ventas diarias consolidadas.
- Las columnas resultantes se renombraron a fecha, solicitante, CIUDAD y cantidad_neta.

VIII-C6. Filtrado de Combinaciones Cliente-Ciudad:

- Se identificaron las combinaciones únicas de solicitante y CIUDAD.
- Solo se seleccionaron aquellas combinaciones que tuvieran al menos 180 días de datos para asegurar una base de datos suficiente para el modelado.

VIII-C7. Definición de la Fecha de Corte:

- Se estableció una fecha de corte (2023-07-01) para dividir los datos en conjuntos de entrenamiento y prueba. La estrategia del proyecto fue usar 18 meses para entrenamiento y 6 meses para prueba.

VIII-C8. Creación de Características (Feature Engineering): Para cada subconjunto de datos (cliente y ciudad), se generaron características temporales y de retraso (lags):

- Lags: Valores de cantidad_neta de los 7 días anteriores (lag_1 a lag_7).
- Medias Móviles: rolling_mean_7 (media móvil de 7 días) y rolling_mean_30 (media móvil de 30 días).
- Tendencia: Una variable de tendencia (trend) basada en la longitud de la serie temporal.
- Variables de Calendario: day_of_week (día de la semana), month (mes), day_of_year (día del año).
- Días Especiales/Festivos: Se añadió una función para incorporar información sobre festivos en Colombia.

Los DataFrames se aseguraron de tener una frecuencia diaria, rellenando los días faltantes con 0 en cantidad_neta.

VIII-C9. Bucle de Evaluación por Cliente y Ciudad: Se iteró sobre cada combinación filtrada de cliente y ciudad.

- División en Entrenamiento y Prueba: Los datos se dividieron en conjuntos de entrenamiento y prueba (X_train, X_test, y_train, y_test) utilizando la fecha_corte.
- Validación de Longitud Mínima: Se incluyó una condición para saltar el entrenamiento si los conjuntos de entrenamiento tenían menos de 180 días o los de prueba menos de 30 días.
- Escalado de Datos: Los datos de entrenamiento y prueba (X_train, X_test) se escalaron utilizando StandardScaler.

VIII-C10. Entrenamiento y Apilamiento de Modelos:

- Modelos Base: Se entrenaron un modelo Random Forest Regressor y un modelo XGBoost Regressor de forma independiente en los datos de entrenamiento escalados (X_train_scaled, y_train).
- Predicciones Base: Se obtuvieron las predicciones (rf_pred, xgb_pred) de ambos modelos base sobre el conjunto de prueba escalado (X_test_scaled).
- Meta-Modelo (Stacking): Las predicciones de los modelos base (rf_pred, xgb_pred) se combinaron (np.column_stack) para formar el stacked_input. Un modelo Ridge Regressor (Ridge(alpha=1.0, random_state=42)) fue entrenado como el meta-modelo sobre este stacked_input y los valores reales del conjunto de prueba (y_test) para generar la final_pred.

VIII-C11. Evaluación del Modelo Apilado: Se calcularon y registraron varias métricas para evaluar el rendimiento del modelo apilado:

- MAE (Mean Absolute Error): Para medir el error absoluto promedio.
- Promedio de Ventas Reales en el conjunto de prueba.
- MAE Relativo: MAE dividido por el promedio de ventas reales.
- Accuracy, Precision, Recall y F1-score: Calculadas como 1 - (MAE / Promedio_real) y sus derivadas.
- MAPE (Mean Absolute Percentage Error): Se calculó específicamente para valores reales distintos de cero para evitar divisiones por cero. El objetivo del proyecto era reducir el MAPE a menos del 10 %.

Los resultados se almacenaron en una lista resultados.

VIII-C12. Visualización y Guardado de Resultados: Se generaron y guardaron gráficos de series temporales que comparaban las ventas reales con las predicciones, así como gráficos de dispersión (scatter plots) de las predicciones frente a las ventas reales para cada cliente y ciudad. Finalmente, todos los resultados de la evaluación se consolidaron en un DataFrame y se guardaron en un archivo Excel llamado resultados_modelo_stacked_random_forest_xgboost.xlsx.

VIII-C13. Clasificación y Distribución del Desempeño: Los resultados fueron clasificados por su MAPE y Accuracy en categorías de desempeño ('Excelente', 'Buena', 'Aceptable', 'Mala'). Se imprimió y visualizó la distribución porcentual de estas clasificaciones, junto con los promedios de MAPE y Accuracy para el modelo.

VIII-C14. ¿Porque se crearon características para este modelo?: A diferencia de otros modelos de series temporales como SARIMA o Prophet, el modelo apilado que utiliza Random Forest y XGBoost requiere de la creación de características (feature engineering). Esto se debe a que:

- Naturaleza de los modelos: Random Forest y XGBoost son algoritmos basados en árboles de decisión. Aunque son muy efectivos para capturar relaciones no lineales y manejar diversas características, no están diseñados intrínsecamente para entender la secuencia o temporalidad de los datos de una serie de tiempo por sí mismos.
- Diferencia con otros modelos: Modelos como SARIMA o Prophet tienen mecanismos internos que les permiten modelar la estacionalidad, la tendencia y los efectos de días festivos de forma automática. En cambio, para que Random Forest y XGBoost puedan capturar estos patrones temporales, es necesario transformar explícitamente la información de la serie temporal en características numéricas que puedan ser .ªprendidas"por estos modelos.

VIII-C14a. ¿Cuáles características se les crearon?: Para el modelo apilado de Random Forest y XGBoost, se generaron diversas características:

- Características de retraso (Lags):
  - lag_1 a lag_7: Representan el valor de la cantidad_neta de los 7 días anteriores.
- Medias móviles:
  - rolling_mean_7: Media móvil de la cantidad_neta de los últimos 7 días.
  - rolling_mean_30: Media móvil de la cantidad_neta de los últimos 30 días.
- Tendencia:
  - trend: Una variable numérica que representa la secuencia o el progreso del tiempo en la serie.
- Variables de calendario:
  - day_of_week: El día de la semana (0 para lunes, 6 para domingo).
  - month: El mes del año.
  - day_of_year: El día del año.
- Días especiales/Festivos:
  - es_festivo: Un indicador booleano que es 1 si la fecha es un día festivo en Colombia para los años 2022 y 2023, y 0 en caso contrario.
  - es_ultimo_dia_mes: Un indicador booleano que es 1 si es el último día del mes.
  - es_primer_dia_mes: Un indicador booleano que es 1 si es el primer día del mes.
  - es_viernes: Un indicador booleano que es 1 si es viernes.
  - es_lunes: Un indicador booleano que es 1 si es lunes.
  - es_fin_semana: Un indicador booleano que es 1 si es fin de semana (sábado o domingo).

VIII-C14b. ¿Por qué se crearon estas características?: Estas características se crearon para permitir que los modelos Random Forest y XGBoost comprendan y capturen los patrones temporales inherentes a los datos de la demanda de huevos, que son cruciales para un pronóstico preciso:

- Lags: Son fundamentales para capturar la autocorrelación de la serie temporal. La demanda de un día está altamente correlacionada con la demanda de días anteriores. Esto permite que el modelo aprenda patrones de repetición diaria o semanal.
- Medias Móviles: Ayudan a suavizar las fluctuaciones y a capturar la tendencia a corto y mediano plazo. Proporcionan una indicación del nivel promedio de demanda reciente, lo que puede ser un predictor fuerte para el futuro cercano.
- Tendencia: Permite al modelo capturar cualquier crecimiento o decrecimiento constante en la demanda a lo largo del tiempo, lo cual es importante para series temporales de largo plazo.
- Variables de Calendario: Son esenciales para modelar la estacionalidad y los patrones cíclicos. La demanda de huevos, al ser un producto perecedero, presenta fluctuaciones estacionales pronunciadas. Por ejemplo, la demanda puede ser diferente en ciertos días de la semana, meses específicos o épocas del año debido a factores culturales o de consumo.
- Días Especiales/Festivos: Los productos perecederos son sensibles a eventos externos y promociones y a festividades locales. Estas variables permiten al modelo identificar cambios atípicos en la demanda asociados a días específicos no capturados por la estacionalidad regular, como festivos o el inicio/fin de mes, que pueden influir en los patrones de compra y consumo.

VIII-C14c. ¿Cómo ayuda eso al modelo final?: La creación de estas características beneficia significativamente al modelo apilado (Random Forest y XGBoost) y al pronóstico final de varias maneras:

- Mejora de la precisión: Al proporcionar a los modelos base (Random Forest y XGBoost) información explícita sobre la temporalidad y los patrones de demanda, estos pueden aprender relaciones más complejas y sutiles en los datos, lo que lleva a predicciones más precisas.
- Captura de patrones complejos: Los modelos Random Forest y XGBoost son efectivos para capturar relaciones no lineales. Con las características de tiempo y de retraso, pueden identificar patrones de autocorrelación, estacionalidad y tendencias que son fundamentales en el pronóstico de series temporales de demanda, especialmente en un producto con volatilidad de la demanda y estacionalidad como los huevos.
- Robustez y rendimiento superior: La literatura destaca que XGBoost ha superado a Random Forest en la predicción de la tasa de producción de huevos debido a su habilidad para capturar patrones sutiles en datos no lineales. Además, en comparaciones para pronosticar alimentos perecederos, XGBoost a veces ha superado a modelos de Deep Learning como LSTM en RMSE a nivel de tienda. Al combinar las fortalezas de ambos en un enfoque de apilamiento (ensemble), se busca aprovechar lo mejor de cada modelo, resultando en un pronóstico más robusto y preciso para un producto altamente perecedero.

- Información para el meta-modelo: Las predicciones de los modelos base (Random Forest y XGBoost), que ya son robustas gracias al feature engineering, se utilizan como entrada para un meta-modelo (Ridge Regressor). Este meta-modelo aprende a combinar las predicciones de manera óptima, lo que ha demostrado mejorar la exactitud en el pronóstico en comparación con el mejor modelo individual.

IX. Evaluación de los modelos

Las métricas de calidad empleadas para la evaluación de los modelos, incluyendo SARIMA, Prophet, y el modelo apilado de Random Forest y XGBoost, fueron las siguientes:

IX-A. Métricas Cuantitativas

- MAE (Error Absoluto Medio): Mide la magnitud promedio de los errores en un conjunto de predicciones, sin considerar su dirección.
- MAPE (Error Porcentual Absoluto Medio): Mide la precisión de las predicciones como un porcentaje del valor real.
- MAE Relativo
- Accuracy (Precisión): Se calculó como $1 - \left(\frac{MAE}{Promedio\_Real}\right)$.
- Precision (Precisión): Se calculó como $1 - \left(\frac{MAE}{Promedio\_Real}\right)$.
- Recall (Sensibilidad/Exhaustividad): Se calculó como $1 - \left(\frac{MAE}{Promedio\_Real}\right)$.
- F1-Score: Una métrica que combina precisión y recall, calculada como $2 \times \left(\frac{Precision \times Recall}{Precision + Recall}\right)$.

IX-B. Clasificación del Desempeño

Para una interpretación más intuitiva de los resultados, las métricas MAPE y Accuracy se clasificaron en categorías de desempeño:

- MAPE:
  - Excelente: MAPE <10 %
  - Buena: MAPE <20 %
  - Aceptable: MAPE <50 %
  - Mala: MAPE >= 50 %
- Accuracy:
  - Excelente: Accuracy >0.9
  - Buena: Accuracy >0.75
  - Aceptable: Accuracy >0.6
  - Mala: Accuracy <= 0.6

IX-C. Evaluación Cualitativa y Visual

- Se realizó un análisis visual de las predicciones versus los valores reales.
- Se generaron gráficas de series temporales para comparar las ventas reales con las predicciones del modelo para cada combinación de cliente y ciudad.
- Se crearon gráficos de dispersión (scatter plots) de las predicciones versus las ventas reales para visualizar la correlación y la distribución de los errores.

X. Comparación de evaluación de los modelos

X-A. Evaluación del Modelo Prophet

X-A1. MAPE (%):

- Excelente (<10 %): 0.77 % de los casos.
- Buena (<20 %): 5.71 % de los casos.
- Aceptable (<50 %): 43.36 % de los casos.
- Mala (>= 50 %): 50.15 % de los casos.
- MAPE Promedio: 221.85 %.

X-A2. Accuracy:

- Excelente (>0.9): 0.77 % de los casos.
- Buena (>0.75): 12.96 % de los casos.
- Aceptable (>0.6): 35.80 % de los casos.
- Mala (<= 0.6): 50.46 % de los casos.
- Accuracy Promedio: 0.4565.

X-B. Evaluación del Modelo Apilado (Random Forest + XGBoost)

X-B1. MAPE (%):

- Excelente (<10 %): 0.45 % de los casos.
- Buena (<20 %): 3.28 % de los casos.
- Aceptable (<50 %): 68.85 % de los casos.
- Mala (>= 50 %): 27.42 % de los casos.
- MAPE Promedio: 52.29 %.

X-B2. Accuracy:

- Excelente (>0.9): 0.15 % de los casos.
- Buena (>0.75): 2.24 % de los casos.
- Aceptable (>0.6): 9.54 % de los casos.
- Mala (<= 0.6): 88.08 % de los casos.
- Accuracy Promedio: 0.2661.

X-C. Evaluación del Modelo SARIMA

X-C1. MAPE (%):

- Excelente (<10 %): 0 % de los casos.
- Buena (<20 %): 0.87 % de los casos.
- Aceptable (<50 %): 31.69 % de los casos.
- Mala (>= 50 %): 67.44 % de los casos.
- MAPE Promedio: 145.94 %.

X-C2. Accuracy:

- Excelente (>0.9): 0 % de los casos.
- Buena (>0.75): 0.29 % de los casos.
- Aceptable (>0.6): 2.76 % de los casos.
- Mala (<= 0.6): 96.95 % de los casos.
- Accuracy Promedio: -0.3038.

X-D. Comparación entre los Modelos

X-D1. ¿Cuál es mejor?: El modelo apilado (Random Forest + XGBoost) es el mejor entre los tres modelos evaluados. Aunque sus porcentajes de .Excelentez "Buena.en MAPE son bajos (0.45 % y 3.28 % respectivamente), su MAPE promedio es significativamente menor (52.29 %) en comparación con Prophet (221.85 %) y SARIMA (145.94 %). Esto indica que, en general, el modelo apilado tiene errores porcentuales absolutos medios mucho más bajos.

X-D2. ¿Cuál es peor?: El modelo SARIMA es el peor de los tres. Presenta el MAPE promedio más alto (145.94 %) y tiene el porcentaje más alto de casos con desempeño "Malo.en MAPE (67.44 %). Además, su Accuracy promedio es negativo, lo que sugiere que las predicciones son extremadamente pobres.

X-D3. ¿Difieren mucho entre ellos?: Sí, difieren mucho. Las diferencias en el MAPE promedio son sustanciales: el modelo apilado (52.29 %) es significativamente mejor que SARIMA (145.94 %) y Prophet (221.85 %). En términos de MAPE .Excelente"(el objetivo del proyecto de menos del 10 %), todos los modelos tienen porcentajes muy bajos (<1 %). Sin embargo, el modelo apilado logra que un porcentaje mucho mayor de casos se clasifique como .Aceptable.en MAPE (68.85 %) en comparación con Prophet (43.36 %) y SARIMA (31.69 %). En cuanto a la Accuracy, el promedio del modelo apilado (0.2661) es bajo, pero mucho mejor que el de SARIMA (-0.3038). Prophet tiene la mejor Accuracy promedio (0.4565).

X-D4. ¿Vale la pena seguir trabajando con ellos o se debería descartar alguno?: El objetivo del proyecto es reducir el error de pronóstico a menos del 10 % (MAPE). Ninguno de los modelos logra consistentemente este objetivo, ya que el porcentaje de casos con MAPE .ˣᵉcelente.ᵉˢ muy bajo para todos (menos del 1 %).

- SARIMA debería ser descartado o su enfoque reevaluado debido a su bajo rendimiento general y a los resultados negativos en Accuracy.
- Prophet, aunque mejor que SARIMA, también muestra un MAPE promedio muy alto y un alto porcentaje de desempeño "Malo". Si bien es un modelo adecuado para series temporales de negocios con múltiples estacionalidades y días festivos, sus resultados actuales sugieren que podría necesitar un ajuste más profundo o que la complejidad de la demanda de huevos lo supera.
- El modelo apilado (Random Forest + XGBoost) es el que presenta la mejor perspectiva para seguir trabajando. Aunque no cumple completamente el objetivo del 10 % de MAPE, es el que más se acerca al objetivo en términos de MAPE promedio y el que tiene la mayor proporción de casos con un desempeño .ᴬceptable". Se debería continuar trabajando con este modelo, buscando optimizaciones adicionales, como:
  - Refinar la ingeniería de características: Se podrían explorar características adicionales o diferentes formas de incorporar la temporalidad y los eventos externos que afectan la demanda de huevos, que es un producto con volatilidad y estacionalidad significativas.
  - Ajuste de hiperparámetros (Hyperparameter Tuning): Optimizar los hiperparámetros de Random Forest y XGBoost, así como los del meta-modelo (Ridge Regressor).
  - Explorar otros meta-modelos: Si bien Ridge Regressor se usó en el apilamiento, la literatura sugiere que la combinación óptima de modelos puede mejorar la exactitud.
  - Análisis por segmento: Aunque se filtra por cliente y ciudad, podría ser útil analizar el desempeño en diferentes segmentos de demanda (por ejemplo, clientes mayoristas vs. minoristas, tipos de huevo).

XI. Análisis de de complejidad de los modelos

Este análisis de complejidad computacional de los tres modelos: Prophet, Random Forest + XGBoost apilados, y SARIMA, incluyen los tiempos de entrenamiento e inferencia con estadísticas como promedio, mediana, máximo y mínimo.

XI-A. Modelo Prophet

Para 648 modelos evaluados:

- Tiempos de Entrenamiento:
  - Tiempo Total: 96.92 segundos (1.62 minutos).
  - Promedio por modelo: 0.1496 segundos.
  - Mediana por modelo: 0.1444 segundos.
  - Máximo por modelo: 0.7532 segundos.
  - Mínimo por modelo: 0.1199 segundos.
- Tiempos de Inferencia:
  - Tiempo Total: 41.31 segundos (0.69 minutos).
  - Promedio por modelo: 0.0637 segundos.
  - Mediana por modelo: 0.0564 segundos.
  - Máximo por modelo: 1.0370 segundos.
  - Mínimo por modelo: 0.0347 segundos.

XI-B. Modelo Apilado (Random Forest + XGBoost)

Para 671 modelos evaluados:

- Tiempos de Entrenamiento:
  - Tiempo Total: 224.44 segundos (3.74 minutos).
  - Promedio por modelo: 0.3345 segundos.
  - Mediana por modelo: 0.3152 segundos.
  - Máximo por modelo: 0.8244 segundos.
  - Mínimo por modelo: 0.1798 segundos.
- Tiempos de Inferencia:
  - Tiempo Total: 5.05 segundos (0.08 minutos).
  - Promedio por modelo: 0.0075 segundos.

- Mediana por modelo: 0.0073 segundos.
- Máximo por modelo: 0.0236 segundos.
- Mínimo por modelo: 0.0000 segundos.

XI-C. Modelo SARIMA

Para 688 modelos evaluados:

- Tiempos de Entrenamiento:
  - Tiempo Total: 521.95 segundos (8.70 minutos).
  - Promedio por modelo: 0.7586 segundos.
  - Mediana por modelo: 0.7643 segundos.
  - Máximo por modelo: 1.7629 segundos.
  - Mínimo por modelo: 0.1491 segundos.
- Tiempos de Inferencia:
  - Tiempo Total: 3.68 segundos (0.06 minutos).
  - Promedio por modelo: 0.0053 segundos.
  - Mediana por modelo: 0.0050 segundos.
  - Máximo por modelo: 0.0222 segundos.
  - Mínimo por modelo: 0.0000 segundos.

XI-D. Conclusión del análisis de complejidad

Comparativamente, se observa que SARIMA tuvo los tiempos de entrenamiento más largos en promedio y en total, mientras que los modelos apilados de Random Forest y XGBoost, así como SARIMA, mostraron tiempos de inferencia significativamente más bajos que Prophet.

XII. Conclusiones

El objetivo principal del proyecto era desarrollar y evaluar un modelo de Machine Learning capaz de pronosticar la demanda de huevos con una precisión superior, buscando reducir el error a menos del 10 % (MAPE).

XIII. Desempeño de los Modelos Evaluados

XIII-A. Modelo Prophet

- Demostró un desempeño .$^{\text{Ex}}$celente"(MAPE $<$10 %) en solo el 0.77 % de los casos evaluados.
- Tuvo un desempeño "Buena"(MAPE $<$20 %) en un 5.71 % de los casos, y .$^{\text{A}}$ceptable"(MAPE $<$50 %) en un 43.36 %.
- Sin embargo, el 50.15 % de los casos fue clasificado como "Mala"(MAPE $>=$ 50 %).
- El MAPE promedio fue de 221.85 %, y el Accuracy promedio de 0.4565.

XIII-B. Modelo Apilado (Random Forest + XGBoost)

- Aunque mostró un porcentaje bajo de casos .$^{\text{Ex}}$celente"(MAPE $<$10 %) en un 0.45 %, y "Buena.$^{\text{en}}$ un 3.28 %.
- Un 68.85 % de los casos fue .$^{\text{A}}$ceptable"(MAPE $<$50 %), y el 27.42 % fue "Mala".
- Presentó el MAPE promedio más bajo de los tres, con 52.29 %.
- El Accuracy promedio fue de 0.2661.
- Fue interesante ver que al probar regresion lineal y regresor Ridge como meta-modelos, las métricas cuantitativas dieron exactamente igual.

XIII-C. Modelo SARIMA

- Tuvo el rendimiento más limitado en comparación con los otros modelos.
- No alcanzó el desempeño .$^{\text{Ex}}$celente.$^{\text{en}}$ ningún caso (0 % de los casos con MAPE $<$10 %).
- El 96.95 % de los casos fueron clasificados como "Mala"según la métrica de Accuracy.
- Su MAPE promedio fue de 145.94 %, y su Accuracy promedio fue negativo (-0.3038), indicando un desempeño muy pobre.

XIII-D. Comparación entre los modelos

- El modelo apilado (Random Forest + XGBoost) es el mejor entre los tres modelos evaluados. A pesar de que su porcentaje de casos .Excelente.en MAPE es bajo (0.45 %), su MAPE promedio (52.29 %) es significativamente inferior en comparación con Prophet (221.85 %) y SARIMA (145.94 %), lo que indica errores porcentuales absolutos medios considerablemente más bajos en general. Además, logró que un porcentaje mucho mayor de casos se clasificara como .Aceptable.en MAPE (68.85 %).
- El modelo SARIMA es el peor de los tres, con el MAPE promedio más alto y el porcentaje más elevado de casos con desempeño "Malo.en MAPE. Su Accuracy promedio negativo refuerza que sus predicciones son extremadamente deficientes.
- Los modelos difieren sustancialmente entre sí en términos de MAPE promedio y la distribución de sus clasificaciones de desempeño.

XIII-E. Viabilidad y Próximos Pasos

- Ninguno de los modelos alcanzó consistentemente el objetivo del 10 % de MAPE para todos los casos evaluados. El porcentaje de casos con un MAPE .Excelente.es muy bajo para todos los modelos (<1 %).
- SARIMA debería ser descartado o reevaluar el enfoque que le dimos debido a su bajo rendimiento general.
- Prophet, aunque mejor que SARIMA, también muestra un MAPE promedio muy alto. Sus resultados actuales sugieren que podría necesitar un ajuste más profundo o que la complejidad de la demanda de huevos lo supera.
- Se debería continuar trabajando con el modelo apilado (Random Forest + XGBoost). Es el que más se acerca al objetivo en términos de MAPE promedio y el que tiene la mayor proporción de casos con un desempeño .Aceptable". Las futuras optimizaciones podrían incluir:
  - Refinar la ingeniería de características: Explorar variables adicionales o diferentes formas de incorporar la temporalidad y los eventos externos que afectan la demanda de huevos. Los modelos basados en árboles como Random Forest y XGBoost, a diferencia de SARIMA o Prophet, requieren la creación explícita de características temporales y de retraso para capturar patrones de series de tiempo.
  - Ajuste de hiperparámetros: Optimizar los hiperparámetros de Random Forest y XGBoost, así como los del meta-modelo (Ridge Regressor).
  - Explorar otros meta-modelos: La combinación óptima de modelos ha demostrado mejorar la exactitud del pronóstico.

Referencias

[1] Policía Nacional de Colombia, "Estadística Delictiva," [Online]. Available: https://www.policia.gov.co/estadistica-delictiva. [Accessed: Feb. 17, 2025].

[2] Policía Nacional de Colombia, "Resultados Operativos," [Online]. Available: https://www.policia.gov.co/resultados-operativos. [Accessed: Feb. 17, 2025].

[3] Fiscalía General de la Nación, "Estadísticas," [Online]. Available: https://www.fiscalia.gov.co/colombia/gestion/estadisticas/. [Accessed: Feb. 17, 2025].

[4] Fiscalía General de la Nación, "Delitos," [Online]. Available: https://www.fiscalia.gov.co/colombia/gestion/estadisticas/delitos/. [Accessed: Feb. 17, 2025].

[5] Observadores Col, "Informe del Delito en Colombia, agosto 2024," [Online]. Available: https://observadorescol.org/wp-content/uploads/informe-del-delito-en-colombia-agosto-2024.pdf. [Accessed: Feb. 17, 2025].

[6] RedAM, "Datos de Delitos," [Online]. Available: https://www.redam.co/ultimos-datos-de-delitos-en-colombia/. [Accessed: Feb. 17, 2025].

[7] Encuesta de Percepción Ciudadana 2018, [Online]. Available: https://2022.dnp.gov.co/programa-nacional-del-servicio-al-ciudadano/Paginas/Encuesta-de-PercepciÃşn-Ciudadana-.aspx. [Accessed: Feb. 17, 2025].

[8] Encuesta de Percepción Ciudadana al Plan Nacional de Desarrollo 2022-2026, [Online]. Available: https://anda.dnp.gov.co/index.php/catalog/158/datafile/F28. [Accessed: Feb. 17, 2025].

Anexos

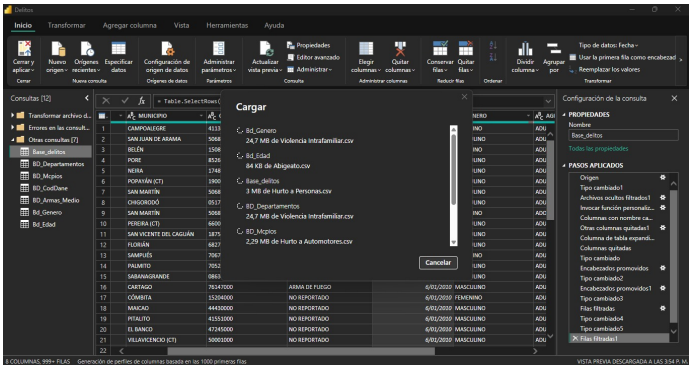

Figura 1. Procesamiento de datos Power BI 1

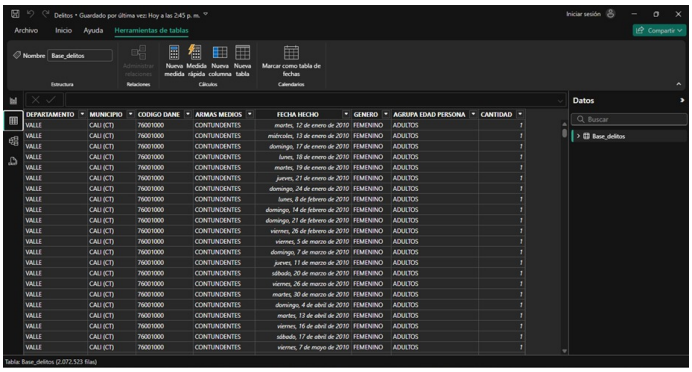

Figura 2. Procesamiento de datos Power BI 2

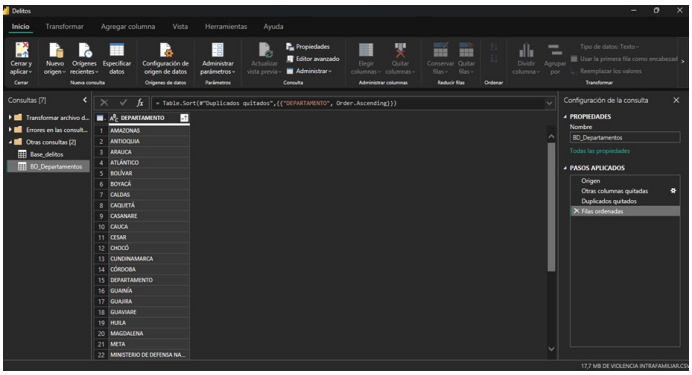

Figura 3. Procesamiento de datos Power BI 3

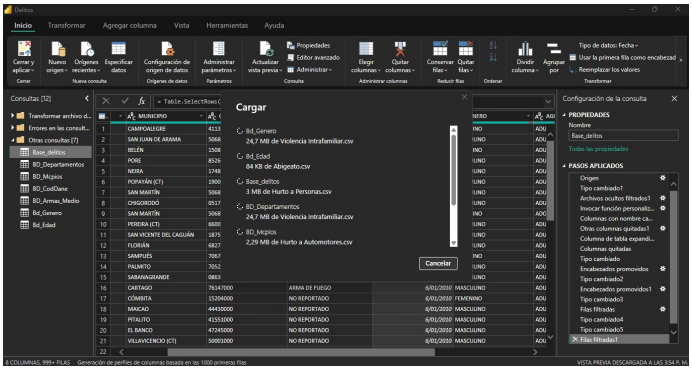

Figura 4. Procesamiento de datos Power BI 4

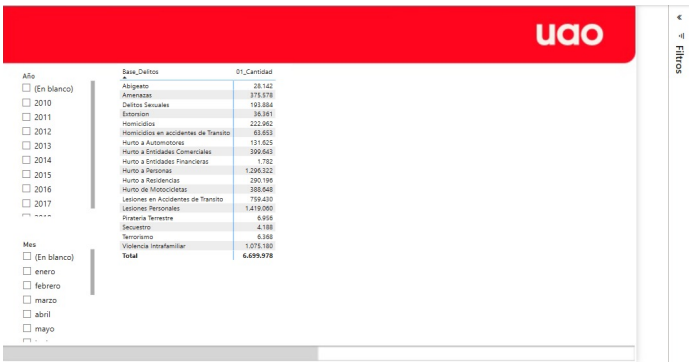

Figura 5. Procesamiento de datos Power BI 5

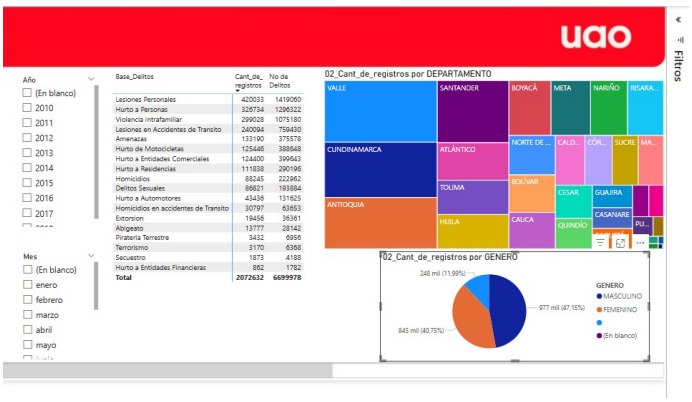

Figura 6. Procesamiento de datos Power BI 6

Agrupación y conversión a CSV

```python
import pandas as pd
import glob
import os

# Ruta de la carpeta origen de los archivos Excel
ruta_origen = r"C:\Users\djseb\Desktop\Delitos Policia\Violencia Intrafamiliar"

# Nombre de la carpeta de origen (para nombrar el CSV)
nombre_carpeta = os.path.basename(ruta_origen)

# Ruta de los archivos Excel
ruta_archivos = os.path.join(ruta_origen, "*.xlsx")

# Ruta de la carpeta destino
ruta_destino = r"C:\Users\djseb\Desktop\Delitos Policia\CSV Extraidos"

# Asegurar que la carpeta destino exista
os.makedirs(ruta_destino, exist_ok=True)

# Lista para almacenar los DataFrames
dataframes = []

# Leer todos los archivos Excel en la carpeta
for archivo in glob.glob(ruta_archivos):
    # Cargar todas las hojas del archivo en un diccionario de DataFrames
    xls = pd.ExcelFile(archivo)

    for sheet in xls.sheet_names:
        df = pd.read_excel(xls, sheet_name=sheet, skiprows=0)  # Asegura que no salte filas

        if not df.empty:
            # Eliminar columnas completamente vacías
            df = df.dropna(how='all', axis=1)

            if not df.empty:
                # Determinar la última columna con datos
                ultima_columna = df.columns[-1]

                # Eliminar filas donde la última columna sea NaN
                df = df.dropna(subset=[ultima_columna])

                # Agregar al conjunto de datos
                dataframes.append(df)

# Unir todos los DataFrames en uno solo
df_final = pd.concat(dataframes, ignore_index=True)

# Nombre del archivo CSV
nombre_csv = f"{nombre_carpeta}.csv"
ruta_csv = os.path.join(ruta_destino, nombre_csv)

# Guardar el DataFrame en un archivo CSV
df_final.to_csv(ruta_csv, index=False, encoding="utf-8")

print(f"Proceso finalizado. Archivo CSV generado: {ruta_csv}")
```

conv.py

No se anexan los scripts de los scrappers dado que los codigos son muy largos.