Evaluación de contaminantes atmosféricos en series de tiempo: Aplicación de modelos de Deep learning

Estudiante: Juan José García Londoño

1. Introducción.

En este proyecto se utiliza la competencia de Kaggle "YDL Air Pollution", cuyo objetivo es predecir la concentración horaria media de hidrocarburos no metánicos (NMHC, µg/m³) a partir de variables medidas por sensores y/o condiciones ambientales.

Se plantea este problema como una tarea de regresión supervisada y se comparan modelos clásicos (baseline) con modelos de deep learning basados en series temporales.

2. Objetivos.

2.1. Objetivo general.

Evaluar un modelo de deep learning que prediga la concentración horaria de NMHC, comparando su desempeño con al menos un modelo clásico de regresión.

2.2. Objetivos específicos.

1) Explorar y preprocesar el dataset de la competencia (limpieza, manejo de NAs, normalización).
2) Construir un modelo baseline (p.ej. regresión lineal o Random Forest).
3) Diseñar e implementar un modelo LSTM/GRU (o CNN 1D) para series temporales.
4) Comparar ambos modelos usando la métrica de la competencia (p.ej. MAE/RMSE) y analizar resultados.

3. Descripción de base de datos (01_exploracion_datos).

La base de datos usada fue descargada por medio de Kaggle en la competencia desarrollada por Mikhail Shkorin, titulada "YDL Air Pollution" cuyo objetivo es predecir la concentración de hidrocarburos no metánicos (NMHC) a partir de variables de calidad del aire y meteorológicas, medidas en una estación ubicada en una zona urbana durante 4 meses, tomando datos cada hora. El archivo denominado "train" en formato CSV (valores separados por comas), incluye 6218 observaciones para 13 variables numéricas.

La competencia establece una variable objetivo, la cual es NMHC(GT), indicando concentración horaria de hidrocarburos no metánicos ($\mu g.m^{-3}$).

Las variables predictoras (features) son:

- ✓ Gases de contaminación: CO(GT), C_6H_6(GT), NO_x(GT), NO_2(GT).
- ✓ Salidas de sensores: PT08.S1(CO), PT08.S3(NO_x), PT08.S4(NO_2), PT08.S5(O_3).
- ✓ Variables meteorológicas: temperatura (T), humedad relativa (RH) y humedad absoluta (AH).

- ✓ Variable temporal: Datetime, usada para ordenar la serie y construir las ventanas temporales.

La columna “Unnamed: 0”, actuaba como un índice heredado. Así, fue eliminada por no contener información relevante.

4. Depuración y limpieza de datos (02_preprocesado).

4.1.Tratamiento de la marca temporal

La variable Datetime se convirtió a tipo fecha-hora y se utilizó para ordenar cronológicamente el conjunto de datos, lo cual es la base para realizar la construcción de ventanas de entrada para los modelos recurrentes (LSTM/GRU). Esta variable no fue usada como variable explicativa, sin embargo, permitió respetar la estructura de serie temporal en todos los splits de entrenamiento, validación y test.

4.2. Valores faltantes y datos inválidos

En el base de datos, el valor -200 se usa como código de medición inválida para varias variables de gases (CO(GT), NMHC(GT), NOx(GT) y NO_2(GT)). Estos valores no representan concentraciones reales, por lo que se reemplazaron sistemáticamente por NaN para su tratamiento como datos faltantes.

Posteriormente se realizaron los siguientes pasos para depuración de datos:

1) Filtrar filas donde no se midio variable respuesta, NMHC(GT). Por lo tanto, se eliminaron todas las filas con NaN en la columna de la variable respuesta. Así, se asegura que el conjunto de datos final solo contiene observaciones donde se mida la variable respuesta. Esta condición es necesaria para aplicar modelos de machine learning.
2) A las variables de entrada se aplicó un imputador por mediana (SimpleImputer(strategy='median')). Así, se le aporta robustez a los datos frente a valores extremos y mantiene la escala original de cada variable.
3) Tras la imputación se verificó que no quedaran valores faltantes en las columnas utilizadas como entrada de los modelos.

4.3. Normalización para modelos de deep learning.

Desde la matriz de las variables predictoras se realizo un escalado a las observaciones con el fin de aplicar correctamente los modelos de Deep learning, los cuales son sensibles a las escalas de las variables. Esta transformación se implementó con StandardScaler y se utilizó para generar las entradas de los modelos LSTM y GRU.

4.4. Preparación de datos para modelos tabulares y secuenciales.

A partir del preprocesado anterior se generaron dos vistas del conjunto de datos:

1) Vista tabular (X_tab, y_tab):
   Corresponde al conjunto de datos imputado (sin escalado obligatorio), donde cada fila corresponde a una hora y contiene todas las variables predictoras. Esta representación se empleó para entrenar los modelos baseline (Regresión Lineal y Random Forest).
2) Vista secuencial (X_seq, y_seq) para redes recurrentes: Se construyeron ventanas deslizantes de longitud 24 horas (window_size = 24) sobre las features escaladas. Así, cada muestra de entrada tiene dimensión (24, n_features) e incluye las últimas 24 observaciones de todas las variables predictoras. Adicionalmente, la etiqueta asociada y_t es la concentración de NMHC(GT) en la hora siguiente a la ventana. Así, el problema se formula como una tarea de predicción un paso adelante en una serie temporal multivariada.

Finalmente, el conjunto secuencial se dividió respetando el orden temporal en tres subconjuntos: aproximadamente 70% para entrenamiento, 15% para validación y 15% para test. Esta partición se mantuvo fija en todas las iteraciones de modelado para permitir una comparación justa entre arquitecturas.

5. Modelos aplicados.

5.1. Modelos baseline (03_modelo_baseline).

Como punto de partida se evaluaron modelos clásicos de regresión sobre la vista tabular de los datos (X_tab, y_tab). El objetivo de estos modelos es establecer una línea base contra la cual comparar posteriormente las arquitecturas de deep learning.

5.1.1. Regresión lineal múltiple.

Este modelo trabaja con la suposicion de que existe una relación lineal entre las variables respuesta, NMHC(GT) y las variables predictoras. Por lo tanto, no determina interacciones no lineales. Tiene facilidad de interpretar y no requiere un fuerte procesamiento de datos. Se utilizó la implementación estándar de LinearRegression de scikit-learn, sin regularización explícita.

5.1.2. Random Forest.

Conjunto de árboles de decisión entrenados sobre bootstraps de los datos y con selección aleatoria de features por árbol. Este modelo de clasificación permite capturar relaciones no lineales y ciertas interacciones entre variables de entrada. Se empleó RandomForestRegressor con un número moderado de árboles (por ejemplo 300) y parámetros por defecto razonables (profundidad libre, n_jobs=-1 para aprovechar CPU). Este modelo resultó ser el mejor

baseline y se utilizó como referencia principal en la comparación con los modelos recurrentes.

Los baselines se entrenaron utilizando el mismo split de entrenamiento/validación/test descrito en la sección de datos, y sus métricas se recogen en la sección de Resultados.

5.2. Modelos de deep learning basados en redes recurrentes (04_modelo_deep_learning).

Dado el carácter temporal de los datos (serie horaria multivariada), se optó por modelos de tipo RNN de la unidad 5 del curso, concretamente redes LSTM y GRU.

Las entradas a estos modelos son tensores de dimensión (T, F), donde:

- ✓ T, es la longitud de la ventana temporal (24 horas anteriores).
- ✓ *F,* es el número de variables predictoras (gases, sensores y variables meteorológicas escaladas).

La salida es escalar y corresponde a la concentración de NMHC(GT) una hora después del final de la ventana.

5.2.1. Arquitectura LSTM.

La primera arquitectura probada fue una red LSTM secuencial con la siguiente estructura:

- ✓ Capa de entrada con forma (24, F).
- ✓ Capa LSTM con 64 unidades, return_sequences=False, que resume la dinámica de las 24 horas en un vector de estado oculto.
- ✓ Capa densa intermedia de 32 neuronas con activación ReLU.
- ✓ Capa de salida densa de 1 neurona lineal para producir la predicción de NMHC(GT).

El modelo se compiló con:

- ✓ Optimizador Adam (learning_rate = 1e-3).
- ✓ Función de pérdida inicial MAE y, en una segunda iteración, MSE sobre el target transformado.
- ✓ Métrica de entrenamiento: MAE.

5.2.2. Arquitectura GRU.

Partiendo de los resultados de la LSTM, se probó una segunda arquitectura basada en GRU (Gated Recurrent Unit), manteniendo la misma estructura general:

- ✓ Entrada (24, F).
- ✓ Capa GRU con 64 unidades.
- ✓ Capa densa de 32 neuronas con activación ReLU.
- ✓ Capa de salida lineal de 1 neurona.

Las GRU tienen menos parámetros que las LSTM y, en muchos problemas de series temporales, ofrecen un compromiso favorable entre capacidad de representación y facilidad de entrenamiento. En este trabajo, la arquitectura GRU con target log-transformado fue la que obtuvo las mejores métricas en el conjunto de test.

5.2.3. **Transformación del objetivo y función de pérdida.**

Durante las primeras pruebas con el target en escala original se observó que el modelo tendía a infraestimar los picos de NMHC(GT). El RMSE era significativamente mayor que el MAE, indicando errores muy grandes en episodios de alta concentración. Para mitigar este comportamiento se introdujo una transformación logarítmica del objetivo:

$$y_{log} = log\big(1 + NMHC(GT)\big)$$

Los modelos LSTM y GRU se entrenaron prediciendo y_{log} utilizando MSE como función de pérdida. En fase de evaluación, las predicciones se transformaron de nuevo al espacio original mediante:

$$\hat{y} = e^{y_{log}} - 1$$

y las métricas (MAE y RMSE) se calcularon sobre ŷ y los valores reales de NMHC(GT). Esta estrategia mejoró de forma notable la capacidad de los modelos recurrentes para reproducir tanto los niveles medios como los picos de la serie.

5.2.4. **Estrategia de entrenamiento y validación.**

Para todos los modelos de deep learning se siguió la misma configuración de entrenamiento:

1. Partición temporal:
   - ✓ ~70% de las ventanas para entrenamiento.
   - ✓ ~15% para validación.
   - ✓ ~15% para test, respetando siempre el orden cronológico.
2. Tamaño de lote: 64 ejemplos por batch.
3. Número máximo de épocas: 50.
4. Parada temprana (early stopping):
   Callback que monitoriza la pérdida de validación (val_loss) y detiene el entrenamiento si no mejora durante 5 épocas consecutivas.

Se restauran los pesos de la mejor época registrada.
Esta estrategia permite evitar sobreajuste, reducir el tiempo de entrenamiento y garantizar que las métricas reportadas en el conjunto de prueba correspondan a un modelo seleccionado únicamente en base al rendimiento sobre la validación.

6. Resultados.

6.1. Comparación cuantitativa de modelos.

En la Tabla 1 se resumen las métricas de desempeño (MAE y RMSE) obtenidas en el conjunto de prueba para los distintos modelos evaluados: regresión lineal, Random Forest y redes recurrentes (LSTM y GRU) con y sin transformación logarítmica del objetivo.

model	MAE	RMSE	set
RandomForest	29.12	49.79	test
LinearRegression	56.53	86.44	test
LSTM_log_target	109.79	164.71	test
GRU_log_target	95.29	150.73	test

De forma cualitativa, los modelos baseline clásicos presentan los errores más altos, especialmente la regresión lineal, que no es capaz de capturar adecuadamente la fuerte no linealidad y la variabilidad de las concentraciones de NMHC(GT). El Random Forest mejora parcialmente estos resultados gracias a su capacidad para modelar relaciones no lineales, pero sigue quedando claramente por detrás de los modelos recurrentes. En cuanto a las arquitecturas de deep learning, la primera versión de la LSTM entrenada sobre la variable objetivo en crudo ofreció una mejora moderada respecto a los baselines, pero mostró un comportamiento claramente deficiente en los episodios de alta concentración: el modelo tendía a infraestimar los picos, lo que se reflejaba en un RMSE de prueba sensiblemente mayor que el MAE. La introducción de la transformación logarítmica del objetivo supuso un cambio importante. La LSTM entrenada sobre log(1+NMHC) redujo tanto el MAE como el RMSE en el conjunto de test (MAE ≈ 109 µg/m³, RMSE ≈ 164 µg/m³), indicando una mejor adaptación tanto a valores medios como a valores elevados de la serie. Finalmente, la arquitectura GRU con objetivo log-transformado fue la que alcanzó el mejor desempeño global, con un MAE de test próximo a 95 µg/m³ y un RMSE en torno a 150 µg/m³, lo que supone una reducción notable de error frente al mejor baseline. En términos relativos, la mejora en MAE y RMSE frente al Random Forest es significativa, confirmando que la combinación de modelo recurrente

+ transformación del objetivo resulta más adecuada para este problema de predicción de contaminantes atmosféricos.

6.2. Análisis gráfico del ajuste

Las gráficas de dispersión "real vs predicho" y las series temporales comparadas permiten interpretar mejor el comportamiento de los modelos. En los modelos basados en el target sin transformar, la nube de puntos se concentra cerca de la diagonal para valores bajos y medios, pero las predicciones se quedan claramente por debajo de los valores reales cuando las concentraciones superan determinados umbrales. De forma similar, en la representación temporal, la curva predicha es mucho más suave y no alcanza los máximos observados.

Tras aplicar la transformación logarítmica, las predicciones de la LSTM y, especialmente, de la GRU se ajustan mucho mejor a la dinámica de la serie. La curva predicha sigue de forma razonable la forma de los picos y valles, y la dispersión alrededor de la diagonal se reduce, aunque aún se observan algunos episodios de infraestimación en los máximos más extremos. Esto es coherente con el hecho de que los valores muy altos de NMHC son relativamente escasos en el dataset y, por tanto, difíciles de aprender incluso para modelos complejos.

6.3.Interpretación de MAE y RMSE en el contexto del problema

Recordando que tanto MAE como RMSE se expresan en las mismas unidades que la variable objetivo (µg/m³ de NMHC), los valores obtenidos indican que, incluso con el mejor modelo (GRU_log_target), el error típico sigue siendo del orden de varias decenas de µg/m³. Esto sugiere que existen factores no observados en los datos (por ejemplo, emisiones puntuales, condiciones meteorológicas no medidas, cambios en el tráfico, etc.) que limitan la precisión alcanzable.

El hecho de que el RMSE sea sistemáticamente mayor que el MAE en todos los modelos indica la presencia de errores grandes en algunas observaciones. Sin embargo, la reducción del RMSE en el modelo GRU_log_target, junto con la mejora visual en la reproducción de los picos, sugiere que la arquitectura está capturando de manera más adecuada los episodios de alta contaminación, que son precisamente los más relevantes desde el punto de vista de la calidad del aire.

6.4. Limitaciones y posibles mejoras

Entre las principales limitaciones del trabajo se encuentran:

- El uso de una única estación y un conjunto de variables relativamente limitado, lo que restringe la capacidad del modelo para explicar ciertas fluctuaciones bruscas.
- La consideración de un horizonte de predicción de solo una hora; extensiones naturales incluirían la predicción multi–paso (varias horas hacia adelante).
- No se ha realizado una búsqueda exhaustiva de hiperparámetros (tamaño de ventana, número de capas, unidades de la GRU/LSTM), por lo que es probable que existan configuraciones que mejoren aún más los resultados actuales.

Como trabajo futuro se propone extender esta metodología a series más largas y a otros contaminantes atmosféricos, así como integrar información de modelos meteorológicos o de inventarios de emisiones para enriquecer el conjunto de variables explicativas.

7. Referencias.