Entrega 2 Proyecto Desarrollo de soluciones

Brayan Sthefen Gomez Salamanca

Juan Sebastian Ordoñez Acuña

Maria Alejandra Rojas Garzon

Hainer Jair Torrenegra Jimenez

Universidad de los Andes

Maestría en Inteligencia Artificial MAIA

Proyecto - Desarrollo de Soluciones

7 de septiembre de 2025

Predicción de la calidad del aire en Risaralda mediante modelos de aprendizaje automático

1. Resumen del problema

La calidad del aire es un factor determinante para la salud pública y el bienestar de la población. En el departamento de Risaralda, los niveles de material particulado (PM10 y PM2.5) han sido motivo de preocupación debido a su impacto en enfermedades respiratorias y cardiovasculares, especialmente en zonas urbanas con alta actividad vehicular e industrial.

El problema identificado consiste en la falta de herramientas que permitan analizar y predecir de forma oportuna los niveles de contaminación del aire, de manera que las autoridades y la ciudadanía puedan tomar decisiones informadas para mitigar riesgos.

La pregunta de negocio que orienta este trabajo es:

¿Es posible construir un modelo de aprendizaje automático que, a partir de datos históricos de calidad del aire, permita estimar y monitorear los niveles de PM10 y PM2.5 en los municipios de Risaralda, facilitando la generación de alertas y la planificación de medidas preventivas?

El alcance del proyecto incluye:

- Análisis exploratorio y preprocesamiento de los datos para identificar patrones espaciales y temporales y corregir posibles errores.
- Entrenamiento y evaluación de múltiples modelos de aprendizaje automático, con registro en MLflow.
- Implementación de una API que exponga el modelo seleccionado.
- Desarrollo de un dashboard interactivo que muestre predicciones y facilite la interpretación de resultados.
- Aplicar los conocimientos adquiridos sobre CI/CD para el despliegue y operación del modelo con plataformas como circleCI y railway

El proyecto se apoya en datos abiertos oficiales del portal datos.gov.co, que contienen 5,047 registros recolectados en estaciones municipales de Risaralda entre 2007 y 2023, con las variables: Municipio, Estación, Fecha, Diámetro aerodinámico (PM10 o PM2.5) y Medición (μg/m³).

2. Modelos desarrollados y su evaluación

● Selección de características

Se trabajó con el dataset **Calidad del Aire Enriquecido obtenido en el EDA (exploración de datos)**, que incluye variables de contexto temporal y de localización:

- **Fecha** (variable temporal de referencia).
- **Municipio, Estación, Diámetro aerodinámico, DíaSemana** (categóricas).
- **Día, Mes, Año** (numéricas).

Además, se generaron **variables derivadas** para capturar la naturaleza secuencial de la serie temporal:

- **Lags del objetivo** (lag_1, lag_7, lag_30) por grupo (Municipio, Estación, Diámetro aerodinámico), de forma que cada registro sólo use información disponible en el pasado.
- **Promedios móviles (rolling means)** con ventanas de 7 y 30 observaciones, también calculados por grupo y desplazados un paso para evitar fuga de información.
- **Variables de calendario** derivadas de la fecha: día de la semana (dow), mes (month) y año (year).

El preprocesamiento se implementó mediante un **pipeline de scikit-learn** con las siguientes transformaciones:

- Numéricas: imputación por mediana y estandarización (StandardScaler).
- Categóricas: imputación por moda y codificación *one-hot* (OneHotEncoder).

Este esquema asegura consistencia entre entrenamiento, validación y despliegue.

● Entrenamiento

Se planteó un **proceso de validación temporal** (expanding window) mediante TimeSeriesSplit para respetar la naturaleza cronológica de los datos.

Se entrenaron y compararon los siguientes algoritmos:

- **Modelos lineales**:
  - *Ridge Regression*: regresión lineal con regularización L2.
  - *ElasticNet*: combinación de regularización L1 y L2.

- **Modelos no lineales**:
  - *Random Forest Regressor*: ensamblado de árboles con bagging.
  - *Support Vector Regressor (SVR, kernel RBF)*: modelo de soporte con kernel gaussiano.

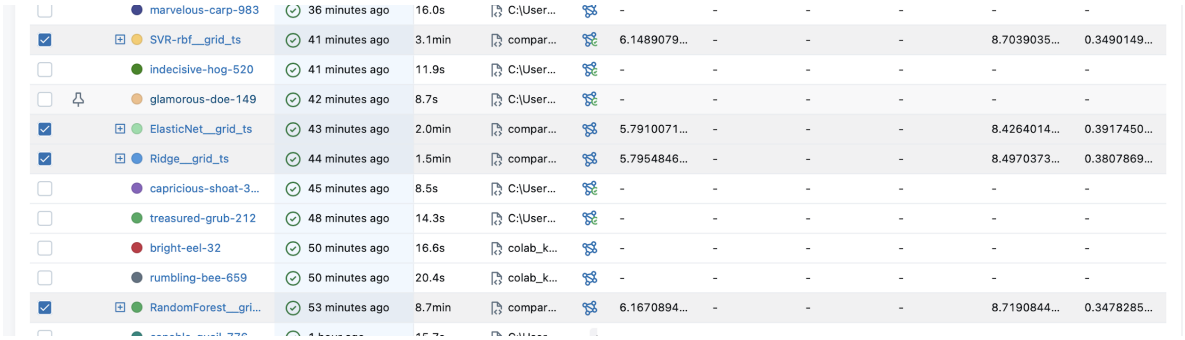
Cada modelo fue entrenado durante los primeros años y evaluado en validación sobre ventanas más recientes, reservando el último año como **holdout final** para medir desempeño fuera de muestra.

• Registro de experimentos en MLflow

Se configuró un **tracking server de MLflow** para almacenar:

- Hiperparámetros explorados en cada grid search.
- Métricas de validación cruzada (cv_rmse, cv_mae, cv_r2).
- Métricas de desempeño en el holdout (holdout_rmse, holdout_mae, holdout_r2).
- Artefactos asociados (modelo serializado, predicciones del holdout).

Cada combinación de hiperparámetros se registró como un *child run*, anidado dentro del *run principal* de cada modelo (ejemplo: ElasticNet__cand_5).



Modelo	**best_cv_rmse**	**best_cv_mae**	**best_cv_r²**	**holdout_rmse**	**holdout_mae**	**holdout_r²**
ElasticNet	8.4264	5.7910	0.3917	**5.9673**	**4.1953**	**0.5709**
Ridge	8.4970	5.7955	0.3809	5.9850	4.2218	0.5684
RandomForest	8.7191	6.1671	0.3478	6.0551	4.4264	0.5582
SVR (RBF)	8.7039	6.1489	0.3490	6.1347	4.3444	0.5465

- **Lectura de los resultados**

  - En validación cruzada (best_cv_*) todos los modelos se comportaron similar (RMSE ≈ 8.4–8.7).
  - En el conjunto de prueba final (holdout):
    - **ElasticNet** obtuvo la mejor combinación de métricas (menor RMSE y MAE, mayor R^2).
    - Ridge estuvo muy cerca, confirmando que los modelos lineales con regularización funcionan bien para este problema.
    - RandomForest y SVR tuvieron un desempeño ligeramente inferior, aunque aún aceptable.

- **Selección del mejor modelo:**

El modelo seleccionado fue **ElasticNet**, con los siguientes hiperparámetros:

- alpha = 0.1
- l1_ratio = 0.8

Estos valores indican que el modelo utiliza una penalización mixta, donde predomina la regularización L1 (80%) sobre L2 (20%). Esto permite obtener un balance entre selección de variables (L1) y estabilidad del modelo (L2).

El desempeño en el conjunto de prueba (holdout) fue:

- **RMSE:** 5.9673
- **MAE:** 4.1953
- **R^2:** 0.5709

Por lo tanto, ElasticNet con esta configuración se considera el modelo final para despliegue en la API.

3. Observaciones y conclusiones sobre los modelos

- **Principales hallazgos**

  - Los cuatro modelos evaluados (**ElasticNet, Ridge, RandomForest y SVR**) presentaron un desempeño similar en validación cruzada (RMSE ≈ 8.4–8.7).

- En el conjunto de prueba final (holdout), el modelo **ElasticNet** logró el mejor rendimiento, con un **RMSE de 5.97**, **MAE de 4.20** y **R^2 de 0.57**.
- Este resultado indica que **los modelos lineales regularizados capturan de forma más eficiente la estructura de los datos** frente a modelos más complejos (RandomForest, SVR).
- El ElasticNet fue configurado con alpha=0.1 y l1_ratio=0.8, privilegiando la regularización L1, lo que ayuda a manejar la multicolinealidad y seleccionar variables relevantes.

- ## Limitaciones
  - **Disponibilidad y calidad de datos**:
    - Se identificaron **saltos temporales en los datos**, es decir, periodos donde no se registraron mediciones. Esto genera interrupciones en la secuencia temporal, lo que dificulta capturar tendencias continuas y reduce la calidad de los lags y promedios móviles generados.
    - No se incorporaron variables externas (ej. meteorología, tráfico, actividad industrial) que podrían mejorar el poder predictivo.
  - **Overfitting**:
    - Aunque no se observó un sobreajuste marcado, la cercanía entre las métricas de validación y holdout muestra que el modelo generaliza de manera estable, pero con margen para mejorar con más datos o features.
  - **Tiempos de entrenamiento**:
    - Los modelos lineales (Ridge, ElasticNet) se entrenaron rápidamente.
    - RandomForest y SVR tuvieron tiempos de entrenamiento mayores sin lograr mejores métricas, lo que refuerza la elección de ElasticNet.

- ## Conclusión

  - El modelo **ElasticNet** se consolida como la mejor alternativa para la predicción de mediciones de calidad del aire en este proyecto, ya que ofrece el mejor equilibrio entre **precisión, interpretabilidad y eficiencia computacional**.

  - En relación con la **pregunta de negocio** —"¿Es posible construir un modelo de aprendizaje automático que, a partir de datos históricos de calidad del aire, permita estimar y monitorear los niveles de PM10 y PM2.5 en los municipios de Risaralda, facilitando la generación de alertas y la planificación de medidas

preventivas?"—, la respuesta es positiva: se logró construir un modelo con un **R^2 cercano al 0.57**, lo cual significa que explica aproximadamente un **57% de la variabilidad** en los datos recientes.

- Esto abre la puerta a **integrar nuevas variables externas** y **ajustar hiperparámetros más finos** en futuras fases, con el fin de aumentar la capacidad explicativa y la utilidad del sistema para la toma de decisiones.

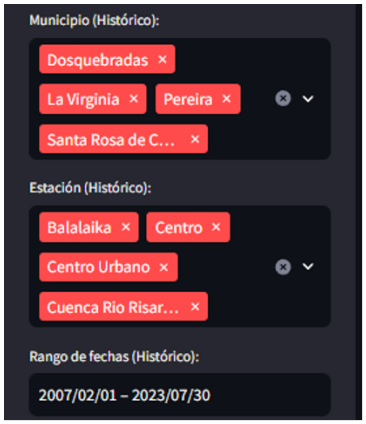
4. Tablero desarrollado

Objetivo del dashboard: el dashboard busca integrar en un solo lugar los datos históricos y las predicciones de calidad del aire (PM10), con el fin de apoyar la toma de decisiones en salud pública, planeación urbana y gestión ambiental.

El dashboard está diseñado para ser **intuitivo, interactivo y visual**, de manera que el usuario pueda navegar fácilmente entre el análisis histórico y las predicciones de calidad del aire. La interfaz combina menús de selección, métricas clave, tablas y gráficas dinámicas que permiten transformar los datos en información útil para la toma de decisiones.

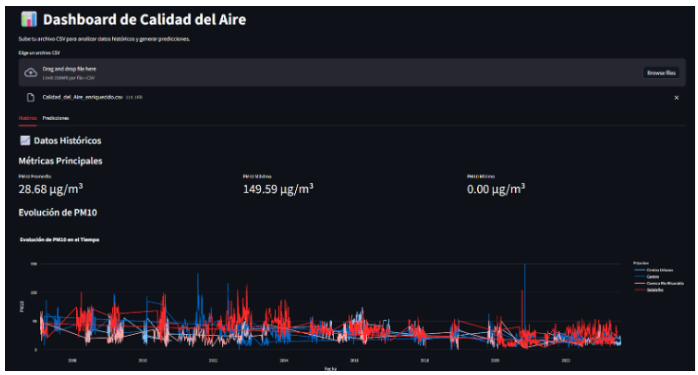
El dashboard se compone de:

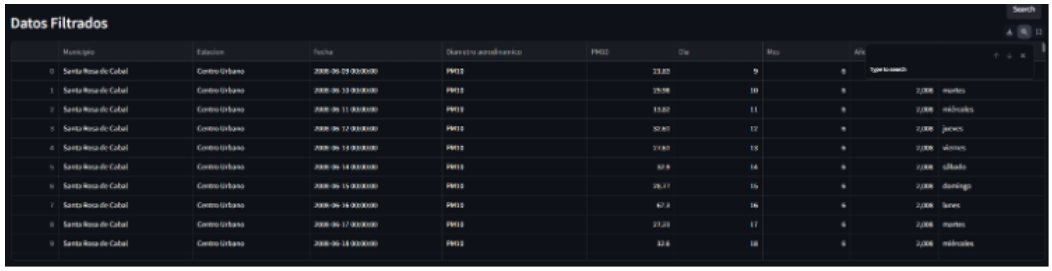
- **Sidebar de Filtros:** se encuentra en el costado izquierdo y contiene los controles de filtrado. Desde allí el usuario puede seleccionar municipios, rangos de fecha y estaciones disponibles en los datos cargados.



- **Pestaña "histórico":** Esta sección está enfocada en el análisis de los datos históricos provenientes del archivo CSV cargado. Se compone de métricas

principales: promedio, máximo y mínimo de PM10, que permiten tener una visión rápida de la calidad del aire en el rango filtrado. Además, incluye una gráfica de evolución en forma de línea, que muestra la variación de PM10 a lo largo del tiempo, diferenciada por estación, lo que facilita identificar patrones y picos de contaminación. Finalmente, se presenta una tabla de datos filtrados, que permite al usuario consultar en detalle las mediciones exactas correspondientes a los filtros aplicados (municipio, estación y rango de fechas).





- **Pestaña Predicciones:** esta sección permite generar estimaciones futuras de PM10 mediante el modelo conectado a la API. Incluye filtros dinámicos de municipio, estación y rango de fechas, un botón de acción para realizar la predicción, una tabla con los resultados estimados y un gráfico de predicciones que facilita la interpretación visual de las tendencias proyectadas.

- **Integración con la API:**

El dashboard se conecta con el backend a través de solicitudes HTTP POST utilizando la librería requests de Python. La dirección del servicio está definida en el archivo principal de Streamlit mediante la variable API_URL = "http://127.0.0.1:8001/api/v1/predict".

Cuando el usuario selecciona municipios, estaciones y un rango de fechas en la pestaña Predicciones, el sistema genera un conjunto de registros en formato JSON. Cada registro contiene los campos que el backend requiere:

- Municipio
- Estacion
- Año, Mes y Dia (extraídos de las fechas seleccionadas)
- DiaSemana (calculado dinámicamente con fecha.day_name())

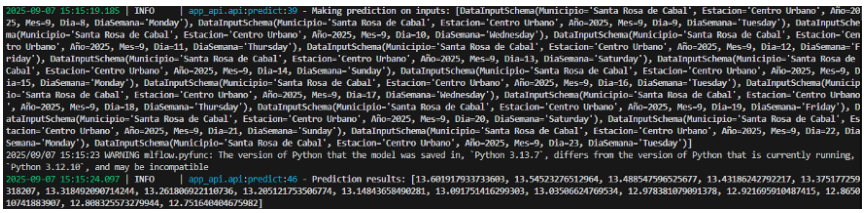
Estos registros se agrupan en una lista bajo la clave "inputs", y son enviados a la API mediante:

response = requests.post(API_URL, json={"inputs": inputs_api})

El backend, implementado con FastAPI, recibe la solicitud, valida la estructura contra un esquema (DataInputSchema) y ejecuta el modelo de machine learning (cargado con MLflow) para obtener los valores predichos de contaminación. La API devuelve la respuesta en JSON bajo la clave "predictions".

El frontend interpreta esta respuesta, construye un DataFrame de pandas con los registros enviados y agrega una nueva columna llamada "Predicción", en la que se asignan los valores devueltos por la API.

Adicionalmente, en la terminal es posible visualizar el registro del flujo de datos enviados al backend, así como la respuesta generada por la API, lo que permite verificar la trazabilidad de la solicitud y la predicción obtenida:



- **Evaluación frente a la pregunta de negocio**:

El dashboard cumple de manera adecuada con el alcance planteado, ya que permite analizar datos históricos de calidad del aire mediante métricas y visualizaciones interactivas, y además realizar predicciones personalizadas a través de la integración del dashboard con la API que expone el modelo entrenado; de esta forma, se responde a la necesidad de contar con una herramienta que combine exploración histórica y capacidad predictiva.

5. Reporte de trabajo en equipo

- **Alejandra – Desarrollo de modelo (MLFlow)**: Pruebas con múltiples modelos, registro de experimentos, selección del mejor modelo.
- **Hainer – Desarrollo de la API (Backend)**: Implementación de la API en FastAPI/Flask, creación de endpoints, despliegue en EC2.
- **Bryan – Desarrollo del Dashboard (Frontend)**: Construcción del tablero interactivo, visualizaciones y documentación de uso.
- **Sebastián – Pruebas, análisis de resultados y documento final**: Validación integral del repositorio, gestión de las actividades, análisis de resultados, redacción del documento final y consolidación del reporte de trabajo en equipo.

Cada integrante del equipo trabajó en una rama individual derivada de main del repositorio, desarrollando las tareas asignadas. Una vez finalizadas, estas fueron integradas, y los cambios pueden visualizarse en el historial del repositorio.