**MEMORIA PROYECTO FINAL MACHINE LEARNING**

# Concurso Kaggle: Predict the Pollutant The Bridge!

Para iniciar la búsqueda del mejor modelo de machine learning que pueda predecir mi contaminante (“*pollutant*”) lo primero que hice fue cargar los datos que me ofrecían en un DataFrame.

Analicé los datos que me ofrecían para poder limpiarlos y así poder trabajar con ellos: vi el tamaño que tenía el DataFrame, nombre de columnas, valores nulos y tipo de los datos.

En cuanto a la columna a predecir (“*pollutant*”) comprobé que, efectivamente, solo tenía los 3 tipos que se indicaban en el enunciado.

Preparación de datos

Como anteriormente ya había hecho una visión global de los datos, ahora procedí a eliminar todas aquellas columnas que no me aportaban nada:

* columna “Unnamed: 0”
* columna “Continent”
* columna “targetRelease”

Al mismo tiempo, la columna a predecir (“*pollutant*”) la codifiqué como indicaban en el enunciado:

* 0 para Nitrogen oxides (NOX)
* 1 para Carbon dioxide (CO2)
* 2 para Methane (CH4)

Ahora que ya tenía los datos limpios, empecé mirando las correlaciones que existían entre las variables numéricas y el target (“pollutant”) y comprobé que la relación entre ellas era prácticamente inexistente.

También miré la correlación entre las distintas variables numéricas entre sí, para ver si había redundancias. Nuevamente vi que no había relación existente salvo en las variables de temperatura (max, min, average) y de viento (max, min, average), que correlacionaban entre ellas.

## Selección de features

El siguiente paso que di fue definir un conjunto X donde cogí todas las variables excepto FacilityInspireID, facilityName, City, DAY, REPORTER NAME, CITY ID, ya que, o bien eran variables categóricas, con más de 10 categorías, lo cual iba a complicar demasiado mi modelo, o eran numéricas, pero sin aporte de valor ninguno.

También definí la variable y con los valores de contaminantes esperados.

A las variables categóricas que había introducido dentro de X les realicé one hot encoding (a través del comando get\_dummies) para poder pasarlas a binaria y así que computacionalmente las pudiera analizar.

Una vez hecho esto separé los datos, tanto de X como de y, en subconjuntos train y test para así poder entrenar el modelo, poder predecir y evaluar cómo de buenas eran las predicciones.

Antes de pasar al siguiente paso, decidí hacer una matriz de Pearson, para ver con colores la importancia de las variables, donde una vez más se corroboraba que no había ninguna relación fuerte entre ellas.

A continuación, decidí entrenar un modelo con un árbol de decisión con los datos que ya había separado previamente, marcando una profundidad de 15 y un mínimo de muestras por hoja de 30. Entrené el modelo, predije y saqué sus métricas de Accuracy y F1 Score.

Aprovechando que ya tenía el árbol de decisión, lo dibujé y saqué las variables más relevantes (“*Feature Importances*”). De aquí saqué como conclusión que las columnas de las industrias y los países tenían una importancia bastante elevada, mientras que el año y el mes no eran demasiado relevantes a la hora de predecir el contaminante.

Decidí modificar las variables del conjunto X para incluir sólo aquellas que habían sido importantes en el árbol de decisión y, con este nuevo conjunto de datos, volver a separarlos en subconjuntos de train y test.

## Buscando el mejor modelo

En un primer intento, al ser un problema de clasificación, pensé en probar con un SVM con kernel lineal. Igualmente, a como hice anteriormente con el árbol de decisión, lo entrené, predije y obtuve sus métricas de Accuracy y F1 Score.

Los resultados que obtuve en Accuracy fueron similares, pero en F1 Score habían mejorado. Recordemos que F1 Score mide el promedio ponderado entre precisión y recall. (Precision es el número de True positives/ (True positives + False positives) y recall es el número de True positives/ (True positives + False negatives).

Viendo estos resultados decidí crear un pipeline para probar varios modelos a la vez.

Los modelos que introduje para entrenar, predecir y comprobar fueron en, primer lugar, **regresión logística**, aplicándole una estandarización, regularización, tanto Lasso, como Ridge, como Elastic Net y ninguna. También le añadí distintos solver, que lo que hace es aplicar diferentes algoritmos a la regresión logística y también una amplia opción de parámetros C (parámetro que aplica regularización con el objetivo de reducir el overfitting), así como diferentes iteraciones.

Otro de los modelos que introduje en el pipeline fue **random forest**, creándolo con una amplia selección de números de estimadores, profundidad de los árboles, mínimo de muestras por hoja y máximo de features.

El último modelo que añadí al pipeline fue **support vector machine (SVM)** donde lo hice añadiéndole el entrenamiento con kernel lineal, polinómico y rbf, así como un amplio rango de parámetro C (parámetro de regularización), coef0, que es el término independiente y solo tiene significado con el kernel polinómico y parámetro gamma, que es el coeficiente del kernel para kernel polinómico y rbf.

En los 3 modelos pedí como métrica para evaluar el modelo el **F1-score macro**.

Después de un tiempo entrenando el modelo me llegó al resultado de que el mejor era el modelo de regresión logística, pero con escasa diferencia con el random forest.

Para ver cómo de bueno era mi modelo realicé una matriz de confusión, una curva ROC AUC y una curva precisión-recall del modelo.

En los 3 gráficos se puede ver bastante claro que la predicción del tipo 2 (Metano) la hace bastante bien, hay muy pocos fallos, pero con la predicción de los tipos 0 y 1 hay bastantes más errores.

Como al hacer el pipeline la diferencia con el segundo mejor modelo era muy pequeña, decido también pasar a estudiarlo un poco más en detenimiento.

El segundo mejor modelo era random forest. Analicé los mismos gráficos que en el modelo anterior: matriz de confusión, curva ROC AUC y curva precisión-recall.

Y cuál es mi sorpresa al comprobar que este modelo predice mucho mejor los contaminantes del tipo 2, aunque predice algo peor los contaminantes de los tipos 0 y 1.

Al subir los datos a Kaggle veo que, aunque mi F1 score era muy parecida, siendo mínimamente mejor con regresión logística, en Kaggle me da un mejor resultado con el modelo de random forest, posiblemente porque el conjunto de datos que utiliza Kaggle para evaluar el modelo contiene más muestras de contaminante 2, lo que explica que para este conjunto de datos precisamente funcione mejor el modelo de random forest y tenga un mejor score en Kaggle.