GRADO EN INGENIERÍA INFORMÁTICA

TRABAJO DE FIN DE GRADO

Predicción de la contaminación atmosférica mediante redes neuronales artificiales

---

Anexo V

Manuales de usuario

Ismael Mira Hernández

4 de julio de 2022

Índice

Introducción

En este anexo se recogen los manuales y guías de usuario para el manejo del software creado a lo largo del proyecto.

En primer lugar, se muestra una guía de la interfaz web, mostrando los aspectos más relevantes de la misma e indicando el significado de los distintos apartados presentes a lo largo de la página, aunque en todo momento se ha buscado que a través de la intuición del usuario se pueda conocer esta información.

Por otro lado, se muestra un manual para la integración del modelo en otros sistemas. Se adjunta la información necesaria para que el usuario pueda obtener la predicción a partir de unas entradas concretas, además de presentar el modelo en varios lenguajes de programación para facilitar la legibilidad del mismo a personas que puedan estar familiarizadas con distintos lenguajes.

Manual de la interfaz web

Al conectarse a la dirección del sitio web, se muestran en la parte superior los datos actuales de Madrid, tanto de contaminación como algunas métricas atmosféricas.

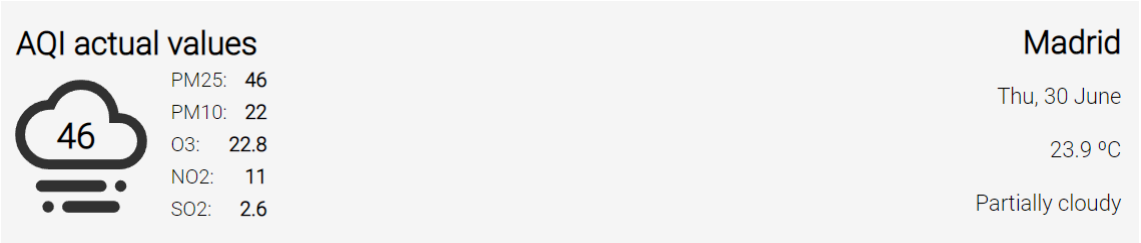

Figura A6. Datos actuales.

En la parte central de la página, se observa una gráfica de barras con los valores del PM2.5. El color de cada barra simboliza el código de colores del índice de la calidad del aire, que se muestra en una leyenda al lado de la gráfica.

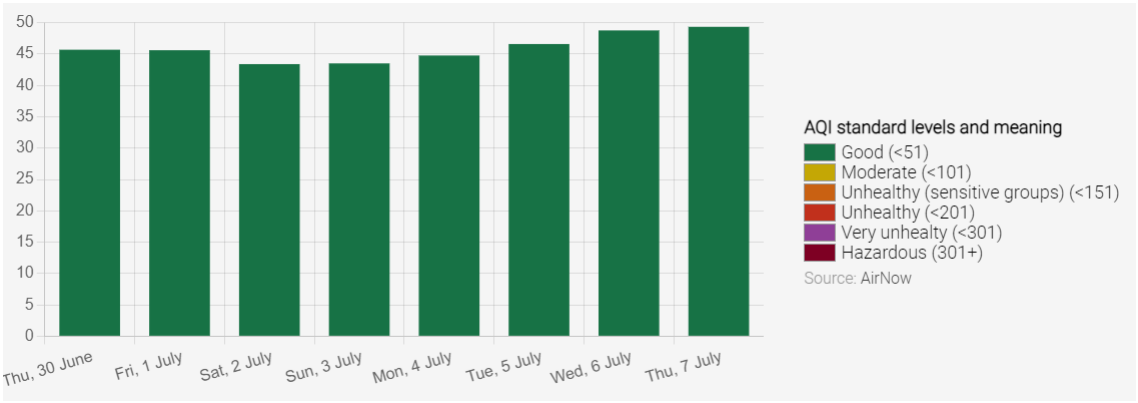

Figura A7. Gráficas y leyenda.

Los botones justo encima permiten mostrar la gráfica del contaminante seleccionado, subrayando y remarcando el activo en ese momento.

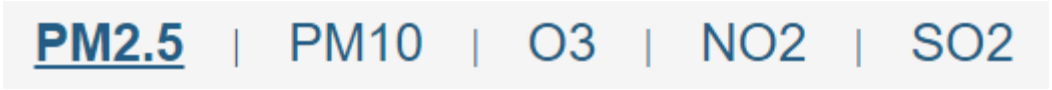

Figura A8. Botones de los contaminantes.

Por último, en la parte inferior se describe brevemente el sistema y el significado de los datos ofrecidos.

Manual para la integración del modelo en otro sistema

El archivo AirPollutionModel.cpp contiene el modelo entrenado con la red neuronal, con los pesos y sesgos calculados tras el entrenamiento y dividido en funciones que asemejan las capas de la red.

```cpp
vector<float> scaling_layer(const vector<float>& inputs)
{
    vector<float> outputs(15);

    outputs[0] = (inputs[0]-15.72999954)/8.804389954;
    outputs[1] = (inputs[1]-6.373929977)/3.488679886;
    outputs[2] = (inputs[2]-3.992089987)/1.996799946;
    outputs[3] = (inputs[3]-54.49259949)/19.50250053;
    outputs[4] = (inputs[4]-24.55690002)/11.86979961;
    outputs[5] = (inputs[5]-32.81600189)/14.54039955;
    outputs[6] = (inputs[6]-24.15929985)/10.28509998;
    outputs[7] = (inputs[7]-3.110419989)/2.051429987;
    outputs[8] = (inputs[8]-1.070829988)/3.709840059;
    outputs[9] = (inputs[9]-15.35890007)/8.133640289;
    outputs[10] = (inputs[10]-21.98600006)/8.937150002;
    outputs[11] = (inputs[11]-8.659509659)/6.911389828;
    outputs[12] = (inputs[12]-1017.650024)/7.242609978;
    outputs[13] = (inputs[13]-10.53100014)/5.43558979;
    outputs[14] = (inputs[14]-58.58539963)/19.50449944;

    return outputs;
}

vector<float> perceptron_layer_1(const vector<float>& inputs)
{
    vector<float> combinations(3);

    combinations[0] = -0.324659 -0.00446057*inputs[0] +0.0845916*inputs[1] +0.
    combinations[1] = 0.267116 -0.00468511*inputs[0] -0.0281986*inputs[1] -0.(
    combinations[2] = 0.346649 +0.0157051*inputs[0] -0.000223656*inputs[1] +0.

    vector<float> activations(3);

    activations[0] = tanh(combinations[0]);
    activations[1] = tanh(combinations[1]);
    activations[2] = tanh(combinations[2]);

    return activations;
}
```

Figura A9. Extracto del modelo en C++.

A partir de este archivo, se pueden calcular las salidas para unas entradas concretas, por ejemplo, pasándolas como argumento al programa o introduciéndolas manualmente.

También podemos traducir este archivo a cualquier otro lenguaje sin necesitar librerías adicionales, simplemente cálculos aritméticos. A continuación, se muestra el mismo extracto de código en Python, donde se aprecian las pequeñas diferencias existentes.

```
def scaling_layer(self,inputs):

    outputs = [None] * 15

    outputs[0] = (inputs[0]-15.72999954)/8.804389954
    outputs[1] = (inputs[1]-6.373929977)/3.488679886
    outputs[2] = (inputs[2]-3.992089987)/1.996799946
    outputs[3] = (inputs[3]-54.49259949)/19.50250053
    outputs[4] = (inputs[4]-24.55690002)/11.86979961
    outputs[5] = (inputs[5]-32.81600189)/14.54039955
    outputs[6] = (inputs[6]-24.15929985)/10.28509998
    outputs[7] = (inputs[7]-3.110419989)/2.051429987
    outputs[8] = (inputs[8]-1.070829988)/3.709840059
    outputs[9] = (inputs[9]-15.35890007)/8.133640289
    outputs[10] = (inputs[10]-21.98600006)/8.937150002
    outputs[11] = (inputs[11]-8.659509659)/6.911389828
    outputs[12] = (inputs[12]-1017.650024)/7.242609978
    outputs[13] = (inputs[13]-10.53100014)/5.43558979
    outputs[14] = (inputs[14]-58.58539963)/19.50449944

    return outputs;


def perceptron_layer_1(self,inputs):

    combinations = [None] * 3

    combinations[0] = -0.324659 -0.00446057*inputs[0] +0.0845916*inputs[1] +0.(
    combinations[1] = 0.267116 -0.00468511*inputs[0] -0.0281986*inputs[1] -0.0
    combinations[2] = 0.346649 +0.0157051*inputs[0] -0.000223656*inputs[1] +0.(

    activations = [None] * 3

    activations[0] = np.tanh(combinations[0])
    activations[1] = np.tanh(combinations[1])
    activations[2] = np.tanh(combinations[2])

    return activations;
```

Figura A10. Extracto del modelo en Python.